Università di Pisa

Facoltà di Ingegneria

Corso di laurea specialistica in
Ingegneria Informatica

Tesi di laurea:

Analisi e sviluppo di un web service per la gestione delle ricette mediche on line

Relatore:
Prof.ssa Gigliola Vaglini

Candidato:
Riccardo Nieto

Correlatore:
Prof. Marco Avvenuti

Anno accademico 2009 – 2010

Sommario

"*Diventiamo ciò che pensiamo di essere*"
Anthony Robbins

1. Introduzione

L'ormai sempre più capillare e invasiva diffusione di Internet ha fatto sì che l'attenzione di aziende e Pubblica Amministrazione si catalizzasse su questo nuovo strumento per ampliare il proprio business o migliorare l'offerta dei servizi.
Di pari passo si è sviluppato una nuova architettura software, conosciuta come SOA (Service Oriented Architecture), la quale essendo orientata ai servizi permette a servizi preesistenti ed implementati con tecnologie eterogenee di comunicare tra loro attraverso l'uso di interfacce.
La SOA fornisce dei concetti puramente teorici, che però trovano la loro applicazione pratica nei Web Service. Un Web Service (cioè un servizio fruibile attraverso la rete) è un sistema software progettato per supportare l'interoperabilità tra sistemi diversi attraverso lo scambio di messaggi basati su XML e l'utilizzo di protocolli Internet. In particolare i Web Service permettono di costruire sistemi distribuiti che sono indipendenti dalla piattaforma tecnologica e dal linguaggio di programmazione utilizzato, il che fa sì che essi svolgano un ruolo principale nello sviluppo e nella realizzazione di architetture B2B (Business-to-Business), dove i servizi forniti da fornitori diversi vengono aggregati per la realizzazione di un nuovo servizio a valore aggiunto. In particolar modo nel mio elaborato di tesi ho utilizzato questa tecnologia per realizzare un servizio che permette la gestione delle ricette mediche direttamente on line, eliminando le difficoltà pratiche del sistema tradizionale.
In particolar modo, grazie all'informatizzazione del sistema, sono eliminati tutti quei problemi di perdita di tempo; da parte del paziente e del medico per ottenere o redigere la ricetta, da parte del farmacista per la conservazione e trattazione delle stesse.

Con la realizzazione di questo progetto non si è avuto la pretesa di implementare un sistema già pronto per l'utilizzo reale in quanto per esempio non si sono trattati i casi particolari presenti nella scrittura delle ricette o è mancata una fase accurata di testing. Quello che si è voluto affrontare con questo elaborato di tesi è una prima esplorazione in ampiezza di questo nuovo campo per capirne le dinamiche e scoprire le problematiche.

2. SOA

In computing, the term service-oriented architecture express a perspective of software architecture that defines the use of ***loosely coupled software services*** *to support the requirements of the business processes and software users. In an SOA environment,* ***resources*** *that can be* ***accessed without knowledge*** *of their* ***underlying platform implementation.***

"Migrating to a service-oriented architecture", IBM DeveloperWorks

Le caratteristiche di base del SOA sono:

1. **Interfacce indipendenti dall'implementazione**: ogni servizio deve essere descritto in maniera indipendente dal linguaggio e dalla piattaforma usata per la sua realizzazione
2. **Reperibilità e binding dinamico**: un servizio deve poter essere ricercato tramite gli elementi dell'interfaccia esposta e richiamato a runtime
3. **Accessibilità da rete**: garantisce la trasparenza rispetto alla locazione
4. **Loosely coupled services**: servizi debolmente accoppiati garantiscono semplicità di gestione
5. **Aggregabilità**: più servizi possono essere composti per portare a termine un particolare task
6. **Gestione dei Fault**: ogni servizio dovrebbe essere in grado di rilevare eventuali errori e notificarli e/o intraprendere azioni correttive

Gli attori di questa architettura sono:

1. **Service Provider**: rende disponibile il servizio, ne espone l'interfaccia e la pubblica su uno specifico registry

2. **Service Requestor/Consumer**: utilizza il servizio tramite l'interfaccia esposta dal provider e ottenuta tramite query a un service registry
3. **Service Registry/Broker**: colleziona le informazioni che si riferiscono ai servizi/provider e le mette a disposizione dei requestor/consumer

Le funzionalità del servizio sono prima una sua descrizione, in cui si illustra le operazioni che il servizio può svolgere, le inquadra in specifici end-point e fornisce le modalità per l'accesso. Dopodiché c'è la possibilità di pubblicare e ricercare delle informazioni sul servizio. A questo punto serve un protocollo per la codifica ed il trasporto dei dati attraverso messaggi.

3. Web Services

3.1 Definizione

Web Services are a software applications identified by a URI, whose interface and bindings are capable of being identified, describe and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols"

Questa è la definizione secondo il W3C dalla quale si evince come i web services siano una particolare incarnazione SOA di cui mantengono le principali proprietà.
I WS garantiscono a dispositivi di natura differente pieno accesso a tipologie di dati eterogenei. Quando due entità si mettono d'accordo per scambiarsi una serie di informazione e per astrarre il procedimento, si affidano ad un sistema in grado di garantire una manutenibilità ed una durata della soluzione il più lunga possibile.
Principalmente un WS espone all'esterno una serie di funzionalità, attraverso un **listener**, anche chiamato server. Un listener è un particolare programma che si mette in ascolto delle richieste che provengono da eventuali client ed ovviamente, cerca di rispondere nel modo migliore.
I WS sono applicabili in qualsiasi tipo di ambiente Web:

1. Internet
2. Intranet
3. Extranet

Sono accessibili da ogni nodo della rete e possono essere combinati in modo da ottenere servizi complessi, semplicemente facendo interagire un insieme di programmi che forniscono ognuno un servizio semplice.

Tutti i servizi comunicano tra loro usando protocolli Web indipendenti dalla piattaforma e dai linguaggi, facilitando l'integrazione di ambienti eterogenei.
Un WS fornisce un'interfaccia programmata application-to-application che può essere invocata da un altro programma, o meglio, da qualsiasi altro tipo di applicazione client o server. Questa interfaccia agisce come un collegamento tra il web e l'applicazione logica attuale che implementa il servizio. Non c'è magia se esiste la possibilità di comunicare tra due programmi scritti in linguaggi diversi, ma solo un intermediario che gestisce le differenze nei tipi di dato. Questo intermediario è una tecnologia basata su XML il quale supporta lo scambio e l'elaborazione di documenti strutturati.

3.2 Vantaggi dei Web Services

Le caratteristiche che distinguono i web services da altri sistemi software distribuiti sono:

- **Open infrastructure**: usano standard e protocolli vendor-independent come HTTP e XML
- **Language transparency**: i web services ed i suoi client possono interoperare anche se sono scritti in linguaggi di programmazione diversi o utilizzando diverse librerie o framemork
- **Modular design**: la loro modularità permette di generare nuovi web service grazie all'integrazione o al layering di quelli già esistenti
- **Firewall policy**: mediante l'uso di HTTP per il trasporto dei messaggi i Web Service non necessitano, normalmente, che vengano effettuate modifiche alle regole di sicurezza utilizzate come filtro sui firewall perché questi hanno già delle policy che ne consentono il traffico sulla porta standard

3.2 Svantaggi dei Web Services

Di rilevanza minore sono gli svantaggi nell'utilizzo di questa tecnologia e sono:

- Attualmente non esistono standard consolidati per applicazioni critiche quali, ad esempio, le transazioni distribuite
- Le performance legate all'utilizzo dei Web Service possono essere minori di quelle riscontrabili utilizzando approcci alternativi di distributed computing quali Java RMI, CORBA, o DCOM
- L'uso dell'HTTP permette ai Web Service di evitare le misure di sicurezza dei firewall (le cui regole sono stabilite spesso proprio per evitare le comunicazioni fra programmi "esterni" ed "interni" al firewall).

3.3 Web Services protocol stack

Il web service protocol stack è uno stack di protocolli di rete il quale è usato per definire, localizzare, implementare e realizzare Web services che possano interagire gli uni con gli altri. Lo stack dei web service ha tipicamente quattro protocolli:

- **Transport protocol**: responsabile per il trasporto dei messaggi tra le applicazioni sulla rete; include HTTP, SMTP, FTP
- **Messaging protocol**: si occupa della codifica dei messaggi un formato comune di tipo XML in modo che possa essere compreso da entrambi i lati della connessione; comprende XML-RPC, WS-Addressing e SOAP
- **Description protocol**: usato per descrivere l'interfaccia pubblica ad uno specifico web service; Tipicamente si usa WSDL.
- **Discovery protocol**: l'idea è centralizzare i servizi in un registro comune in modo che i web service possano pubblicare la loro posizione e descrizione in modo che sia facile scoprire quale servizi sono disponibili sulla rete. UDDI è stato creato per questo scopo, ma non è ancora diffusamente utilizzato

Ulteriori protocolli standard utilizzati sono:

- **WS-Security**: tale standard permette l'autenticazione degli utenti e la confidenzialità dei messaggi scambiati con l'interfaccia del Web Service
- **WS-Reliability**: soddisfa le richieste di quelle applicazioni che richiedono l'invio affidabile di messaggi
- **BPEL**: è un linguaggio per la specifica di processi di workflow che permette di comporre e orchestrare web services al fine di fornire servizi aggregati

La figura seguente mostra lo stack completo.

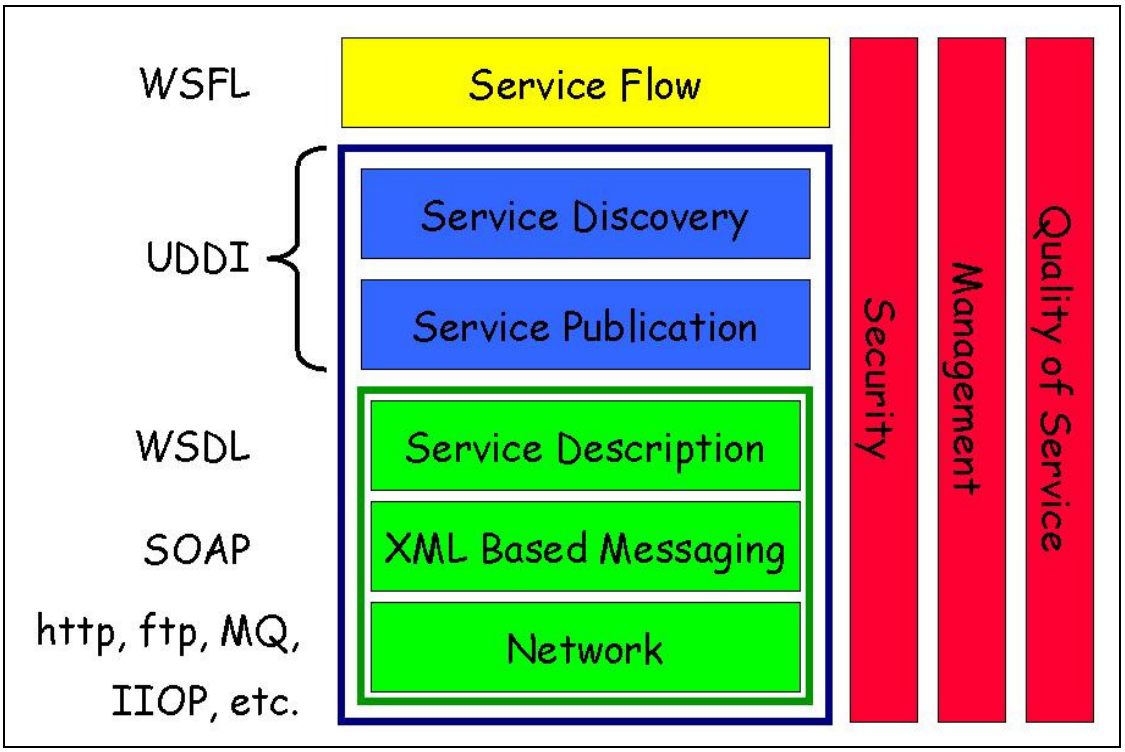

Figura 1: Web Services stack

I protocolli utilizzati per la realizzazione di questo lavoro di tesi sono stati fino ad UDDI escluso in quanto si è ipotizzato un collegamento statico tra client e server del servizio. Tali protocolli verranno presentati nel capitolo seguente.

3.4 Styles of use

I web services sono un insieme di tool che possono essere usati in molte maniere differenti. I tre stili più comuni sono RPC, SOA e REST

- Remote Procedure Call: presenta una interfaccia per la chiamata ad una funzione distribuita; è stato il primo stile utilizzato, ma adesso viene criticato per non essere loosely coupled perché spesso viene implementato mappando i servizi direttamente in un linguaggio specifico
- Service-Oriented Architecture: l'unità base per la comunicazione sono i messaggi invece delle operazioni. Permette un disaccoppiamento maggiore perché il centro del servizio è il "contratto" del servizio, descritto dal WSDL, invece che i dettagli dell'implementazione.
- REpresentational State Transfer: il suo scopo è quello di realizzare l'interazione con risorse stateful invece che con messaggi o operazioni

Lo stile utilizzato nella realizzazione del progetto è stato quello SOA, scelto per la sua semplicità e la sua interoperabilità.
Per creare un web service ci sono due metodologie di sviluppo:

- **Bottom up**: lo sviluppatore prima scrive le classi che implementano il servizio nel linguaggio di programmazione preferito e poi usando dei tool genera il WSDL corrispondente per esportare i metodi del suo web service. Questo, di solito, è l'approccio più semplice
- **Top down**: lo sviluppatore prima scrive il documento WSDL e dopo utilizzando un tool che legge il WSDL crea lo scheletro per le classi che successivamente completerà. Questo è il metodo più difficile, ma crea un design più pulito

Nella mia tesi per realizzare il server ho utilizzato il primo metodo, scrivendo il codice del servizio nel linguaggio di programmazione che ho scelto e successivamente generando in maniera automatica il codice

WSDL che descrive il servizio. A questo punto ho usato il secondo metodo per creare i due client che utilizzano il servizio.

4. Tecnologie utilizzate

4.1 HTTP

In questo capitolo illustrerò brevemente tutte le tecnologie ed i protocolli utilizzati nella realizzazione di questo progetto, illustrando prima lo stack dei web services per poi passare al linguaggio di programmazione ed al web server.

L'HyperText Transfer Protocol è protocollo dello strato dell'applicazione che costituisce il cuore del web. E' importante distinguere fra **applicazioni della rete** e **protocolli dello strato dell'applicazione**. Un protocollo dello strato dell'applicazione è solo un pezzo di un'applicazione della rete. Il protocollo HTTP definisce come i messaggi vengono passati fra browser e server web in particolar modo:

- I **tipi** di messaggi scambiati
- La **sintassi** dei vari tipi di messaggio
- La **semantica** dei campi
- Le **regole** per determinare quando e come un processo invia o risponde ai messaggi

HTTP è usato per trasmettere risorse, non solo file. Una risorsa è identificata da un URI o URL. Un **Uniform Resource Idendifier** è definito secondo il W3C come "il set generico di tutti i nomi/indirizzi che sono stringhe corte che si assegnano alle risorse".

Un **Uniform Resource Locator** è definito come "termine informale" riferito ai protocolli URI più popolari: http, ftp, mailto, etc…

Un URL si compone normalmente di sei parti alcune delle quali opzionali:

protocollo://<username:password@>nomehost<:porta></percorso><?querystring>

- ***protocollo***: descrive il protocollo da utilizzare per l'accesso al server. I protocolli più comuni sono l'HTTP, HTTPS, FTP, MMS ecc. Se il protocollo non viene specificato, generalmente il browser utilizza "HTTP://" come predefinito
- ***username:password@(***opzionale): subito dopo il protocollo, è possibile specificare l'autenticazione (username e password) per l'accesso alla risorsa. Alcuni server consentono di specificare nell'URL le informazioni per l'autenticazione nel formato "username:password@". La sicurezza dell'autenticazione in URL è estremamente rischiosa in quanto le credenziali di accesso vengono inviate al server in chiaro. Inoltre, i server che non necessitano di autenticazione, non considerano in alcun modo i contenuti a sinistra del simbolo "@", questo sistema di autenticazione espone gli utenti al phishing
- ***nomehost***: rappresenta l'indirizzo fisico del server su cui risiede la risorsa. Può essere costituito da un nome di dominio o da un indirizzo IP
- ***porta*** (opzionale): indica al sistema operativo dell'host remoto la porta del processo server al quale affidare la richiesta. Vi è necessità di indicare questo parametro quando il processo server è in ascolto su una porta non conforme allo standard (porta 80 per HTTP)
- ***percorso*** (opzionale): percorso (pathname) nel file system del server che identifica la risorsa
- ***querystring*** (opzionale): se richiesto, al termine dell'url è possibile aggiungere una query string separandola utilizzando il simbolo "?". La query string è una stringa di caratteri che consente di passare al server uno o più parametri. Di norma, la query string ha questo formato: *[...]?parametro1=valore&parametro2=valore2*

HTTP usa come protocollo di trasporto TCP e quindi il client che inizia la comunicazione è responsabile anche di iniziare la connessione TCP. Una volta stabilita la connessione TCP può mandare la richiesta HTTP ed attendere la risposta da parte del server. Mandato il messaggio di

risposta il server chiude la connessione con un ulteriore scambio di segmenti TCP.
Esistono due tipi di connessioni:

- **Non permanente**: usata da HTTP/1.0 nella quale si richiede al client di stabilire una connessione TCP separata per ogni richiesta
- **Permanente**: usata da HTTP/1.1 nella quale il client continua ad usare la connessione TCP esistente per le successive richieste. L'utilizzo di quella permanente richiede collaborazione tra client e server in cui il client richiede una connessione di questo tipo ed il server non deve chiudere la connessione dopo aver mandato la risposta

A sua volta esistono due versioni della connessione permanente:

- **Non incanalata**: il client passa una nuova richiesta solo dopo aver ricevuto dal server la risposta alla precedente
- **Incanalata**: il client può fare richieste consecutive senza attendere le risposte alle precedenti

Il modo di default per HTTP/1.1 usa la connessione permanente con incanalamento.
HTTP usa il modello client-server in cui il client apre una connessione e manda un messaggio di richiesta al server; il server rimanda un messaggio di risposta, contenente solitamente la risorsa richiesta e senza immagazzinare alcuna informazione relativa al client (stateless protocol). Il formato dei messaggi di richiesta e risposta è simile, entrambi sono composti da:

- una linea iniziale
- nessuna o più linee di intestazione (solo *Host* è obbligatoria)
- una linea vuota
- un corpo del messaggio opzionale

La linea iniziale del messaggio di richiesta è una linea composta da tre campi separati da spazi:

- metodo
- URI: indica la risorsa richiesta
- versione HTTP: versione HTTP supportata

Un esempio potrebbe essere: *GET / HTTP/1.1*
Nella tabella seguente un elenco dei metodi previsti dal protocollo.

GET	Serve per recuperare un oggetto dal server passandogli una URL
POST	Come GET, ma in più permette al client ti mandare informazioni al server
PUT	Analogo a POST, permette di mandare dati dal client al server
DELETE	Dà la possibilità di rimuovere oggetti dal server
HEAD	Il server manda solo l'header della risorsa richiesta
TRACE	E' un metodo diagnostico per scoprire il percorso che effettuano i messaggi per arrivare al server
OPTIONS	Usato per scoprire le capacità di un server Web
CONNECT	Riservato esclusivamente ai server intermedi per creare un tunnel verso il server di destinazione

Tabella 1: metodi HTTP

Invece per quanto riguarda la linea iniziale di un messaggio di risposta consiste in una linea di stato composta da tre campi separati da spazi:

- versione HTTP: più alta versione supportata
- codice di stato: codice di stato dell'esito della richiesta
- messaggio di stato: corrispondente al codice di stato

Il codice di stato è un numero di tre cifre, la prima identifica il tipo di risultato e dà una indicazione di alto livello, le altre forniscono dettagli. Nella tabella seguente un elenco delle classi dei codici di stato.

100-199	Messaggio informativo: il server ha ricevuto la richiesta ma un risultato finale non è ancora disponibile
200-299	Successo: il server è stato capace di agire sulla richiesta con successo
300-399	Redirezione: il client deve reinstradare la richiesta a un differente server o risorsa
400-499	Errore sul lato client: la richiesta conteneva un errore che ha impedito al server di agire su di essa con successo
500-599	Errore sul lato server: il server ha fallito ad agire su una richiesta anche se la richiesta sembra essere valida

Tabella 2: codici di stato HTTP

Le intestazioni possono essere di quattro tipi:

1. **generali**
2. **di richiesta**
3. **di risposta**
4. **del corpo dell'entità**

Esse cominciano con un nome seguito dai due punti (:). In alcuni casi è sufficiente solo il nome, ma nella maggior parte dei casi viene inclusa informazione addizionale dopo i due punti; vediamone un esempio:
Intestazione1: valore_1a, valore_1b

4.2 XML

Il World Wide Web Consortium (W3C), in seguito alla guerra dei browser (ovvero la situazione verificatasi negli anni novanta nella quale Microsoft e Netscape introducevano, con ogni nuova versione del proprio browser, un'estensione proprietaria all'HTML ufficiale), fu costretto a seguire le individuali estensioni al linguaggio HTML.

Il W3C dovette scegliere quali caratteristiche standardizzare e quali lasciare fuori dalle specifiche ufficiali dell'HTML. Fu in questo contesto che iniziò a delinearsi la necessità di un linguaggio di markup che desse maggiore libertà nella definizione dei tag, pur rimanendo in uno standard.
Negli ultimi anni XML (eXtensible Markup Language) si è affermato come lo standard de facto per i markup. Fin dalla sua venuta ha trovato spazio nella realizzazione di diverse applicazioni:

- Informazioni di configurazione
- Electronic Data Interchange
- Vector Graphics
- Remote Method Invocation
- Serializzazione di oggetti

XML può essere usato per personalizzare tag che sono rilevanti per il dominio in cui si lavora. La sua forza sta nel rappresentare i dati in un formato testuale indipendente dalla piattaforma.
XML come HTML è un linguaggio di markup che si basa su SGML (Standard Generalized Markup Language) in particolar modo ne è un sottoinsieme.
Come tutti i linguaggi di markup descrive il formato del documento, ovvero il modo in cui il contenuto del documento dovrà essere interpretato.
XML consente di definire elementi personalizzati e quindi è responsabilità del programmatore pianificare l'uso di questi elementi. Nonostante la natura apparentemente libera, i documenti XML sono soggetti a regole ben codificate. Se un documento XML non può essere interpretato con esito positivo da un processore XML allora questi dovrà fornire un messaggio di errore senza cercare di correggere gli errori come invece succedeva per HTML.
I documenti XML sono soggetti a due vincoli:

- **Ben formati**: secondo il W3C
  - Considerato globalmente, è concorde con la produzione etichettata document

- Rispetta tutti i vincoli "well-formedness" espressi in queste specifiche
- Ciascuna entità analizzata, che viene riferita direttamente o indirettamente all'interno del documento risulta ben formata

➢ **Validi**: conformi alla definizione del tipo di documento (DTD) o allo schema usato per specificare la sintassi legale del documento

Un documento XML è composto da due sezioni, un prologo e l'istanza del documento. XML è case sensitive, uno stesso tag assume significato diverso a seconda che sia scritto in maiuscolo o minuscolo. Ogni elemento può contenere altre informazioni rappresentate da coppie nome-valore denominate attributi. Gli attributi non trasportano un'informazione ma descrivono l'informazione trasportata dall'elemento; in tabella sono riportati i tag principali.

Nome	Sintassi	Descrizione
Elementi	*<Tag>content</Tag>*	Tag e contenuto informativo associato
Riferimenti a entità	*&entità_name;*	Servono per associare un'etichetta a un'informazione ripetuta più volte all'interno del documento, in modo da usare l'etichetta invece che l'informazione stessa
Commenti	*<!--commento-->*	Per inserire commenti
Istruzioni	*<?Istruzione?>*	Informazioni per le applicazioni che leggono il documento
Sezioni CDATA	*<![CDATA[.]]*	Inserisce informazioni che non devono essere elaborate secondo la sintassi XML

Tabella 3: tag principali XML

4.2.2 Prologo

Il prologo si trova all'inizio del documento, non deve necessariamente essere presente per avere documenti ben formati, tuttavia il W3C raccomanda di includere almeno la dichiarazione XML; gli altri elementi che lo compongono sono: commenti, istruzioni per l'elaborazione, spazi e dichiarazioni di tipo documento.
La dichiarazione XML ha il seguente formato:
<?xml version = "1.0" standalone = "yes" encoding = "UTF-8"?>

L'elemento *<?xml?>* consente l'uso di tre attributi:

1. *version*: obbligatorio, indica la versione, attualmente l'unica possibile è 1.0
2. *standalone*: opzionale, impostato a "*yes*" se il documento non fa riferimento ad altre entità esterne; in caso contrario "*no*"
3. *encoding*: opzionale, indica la codifica del linguaggio usata per il documento

I commenti possono apparire ovunque in un documento all'esterno di altri markup, ignorati dai parser XML, sono usati per introdurre note esplicative.
<!--A note here-->

Le istruzioni di elaborazione sono direttive per il processore XML. Un'istruzione di elaborazione molto comune e ben compresa è:
<?xml-stylesheet type:"text/css" href="greetings.css"?>

E' importante notare che nel linguaggio XML, gli spazi, i caratteri di ritorno a capo (CR), di avanzamento riga (LF) e di tabulazione sono tutti trattati come spazi.

4.2.3 Istanza del documento

Ciascun documento XML ben formato deve contenere un elemento che possiede al suo interno tutti gli altri elementi. L'elemento che svolge il

ruolo di contenitore è chiamato elemento radice. Esso è una parte molto importante dei documenti XML specialmente quando i documenti sono esaminati dal punto di vista della programmazione, poiché l'analisi dei documenti comincia proprio con l'elemento radice. Le regole da seguire per avere una istanza del documento ben formata sono:

1) Inclusione di uno o più elementi: alla peggio solo l'elemento document
2) Uso dei tag d'inizio e fine per gli elementi non vuoti
3) Chiusura dei tag vuoti con "/>"
4) L'elemento radice deve contenere tutti gli altri elementi
5) Annidamento corretto degli elementi
6) Uso di nomi univoci per gli attributi
7) Uso dei soli cinque riferimenti a entità predefinite
8) Delimitazione dei valori degli attributi con doppi apici
9) Uso di "<" e "&" solo per contrassegnare l'inizio di tag ed entità

4.2.4 Namespace XML

XML offre molta libertà ai programmatori e consente la definizione di tag personalizzati. Al crescere, tuttavia, del numero delle applicazioni XML, è sorto un problema che non era stato previsto dai creatori delle specifiche originali: i conflitti dei nomi dei tag.

La soluzione è usare i namespace. Questi consentono di evitare che un set di tag possa entrare in conflitto con un altro. I namespace sono usati facendo precedere un nome, seguito da due punti, ai tag e ai nomi degli attributi, modificando quindi i nomi per evitare i conflitti.

Un namespace viene creato in questo modo:

<xmlns:book=http://www.book.com/spec>

Il prefisso "book" diventa quindi un identificatore univoco da premettere ad ogni tag appartenente a quel namespace (book:title). L'attributo xmlns può essere usato in qualsiasi elemento, non necessariamente solo nell'elemento radice.

4.3 SOAP

4.3.1 Concetti base

Simple Object Access Protocol è un protocollo di comunicazione che usa XML il quale, basandosi sullo scambio di messaggi unidirezionale e stateless, permette l'integrazione di sistemi eterogenei distribuiti. L'attore della comunicazione che manda il messaggio si chiama SOAP sender, e quello che lo riceve SOAP receiver. Il cammino che il messaggio attraversa dal mittente all'ultimo ricevente si chiama message path. Lungo il cammino il messaggio può attraversare zero o più intermediari. Non tutte le informazioni contenute nel messaggio devono essere destinate all'ultimo destinatario. L'unità atomica di informazione scambiata tra nodi SOAP è il SOAP message, il quale è un documento XML che ha come elemento di root il SOAP envelope. A sua volta questi può avere un SOAP header e deve avere un SOAP body. A loro volta l'header ed il body possono avere unità di informazioni incapsulate al loro interno e questi si chiamano rispettivamente header block o body block. Il SOAP fault è un blocco speciale che contiene le informazioni di errore generate dai nodi.

4.3.2 Messaggio SOAP

Il SOAP Envelope è quindi un documento XML che è scambiato tra le parti e composto da:

- Il nome locale Envelope e l'URI per il namespace *http://www.w3c.org/2001/06/soap-envelope*
- Un elemento figlio con nome Header e stesso URI per namespace
- Il prossimo elemento figlio, obbligatorio, con nome Body e stesso URI per namespace

Il SOAP header invece contiene informazioni utili alla gestione del messaggio come sicurezza, gestione di credenziali, routing, identificativi di transazioni. Grazie all'attributo "actor" si indica il nodo che destinatario di quell'header. Se poi contiene anche *MustUnderstand*

allora si impone al ricevente di generare un errore nel caso non sia in grado di interpretare uno specifico elemento.

<xs:elemento SOAP-ENV: MustUnderstand="true">

L'elemento body contiene il messaggio vero e proprio destinato all'ultimo ricevitore. Poiché le specifiche SOAP definiscono per body il solo figlio fault, è necessario definire un namespace al suo interno. Solitamente tale namespace è associato al file wsdl del web service invocato in quanto gli elementi definiti all'interno di body sono quelli contenuti nel wsdl. Può contenere una richiesta, una risposta o l'indicazione di errore.

```
<SOAP-ENV:Body>
    <m:MetodoXY-Z xmlns:m="provider:Prova"
    SOAP-ENV:encodingStyle="http://..soapencoding">
        <Param1>X</Param1>
        <Param2>Y</Param2>
    </m:MetodoXY-Z>
</SOAP-ENV:Body>
```

Il SOAP fault viene utilizzato per notificare errori e contiene le seguenti parti:

- *<faultcode>:* indicazione di errori interpretabili automaticamente dagli algoritmi
- *<faultstring>:* contiene le stesse informazioni in formato "umano"
- *<faultactor>:* identifica il nodo che ha generato l'errore (URI)
- *<details>:* contiene informazioni dettagliate sugli errori generati durante il processamento dei blocchi body.

4.4 WSDL

Il Web Service Description Language è un linguaggio basato su XML, indipendente dalla piattaforma, utilizzato per descrivere un servizio e le modalità di accesso ad esso. Più precisamente un documento WSDL fornisce informazioni riguardanti l'interfaccia del web service in termini di servizi offerti, di parametri di ingresso/uscita e di possibili eccezioni, l'URL, il protocollo di trasporto ecc…Definisce quindi cosa fa, come comunica e dove si trova nascondendo l'effettiva implementazione del

servizio. In WSDL la descrizione del servizio ha due componenti principali:

- La descrizione funzionale: definisce i dettagli relativi ai dati di input/output, al luogo in cui si trova, al punto di accesso e alla modalità di connessione, esponendo le operazioni messe a disposizione dal servizio
- La descrizione non funzionale: fornisce informazioni secondarie rispetto al messaggio riguardanti le politiche di sicurezza richieste, la qualità e l'affidabilità del servizio

Grazie ai tool di sviluppo oggigiorno nessuno scrive direttamente codice WSDL, ma si limita a generarli o importarli, tuttavia la sua conoscenza è fondamentale per la comprensione ed il miglioramento dei file WSDL creati in automatico dagli strumenti di sviluppo.

4.4.1 Struttura di un file WSDL

Lo standard 2.0 definisce quali sono gli elementi XML che compongono il file, il modo in cui sono correlati e quali sono gli attributi che tali elementi hanno o possono avere. Il documento inizia sempre con un elemento radice chiamato "description" che contiene al suo interno:

- **types**: definisce i tipi dei dati utilizzati nei messaggi che possono essere inviati e ricevuti
- **interface**: descrive le funzionalità astratte fornite dal WS
- **binding**: definisce le informazioni sul protocollo e il formato dei dati relativo ad un particolare elemento interface
- **service**: specifica l'insieme di endpoint relativi al servizio
- **documentation**: contiene informazioni leggibili da un umano relative alle caratteristiche e al funzionamento del WS

Ciascuno di questi elementi identifica una distinta sezione del documento contenente informazioni relative a uno specifico aspetto.

Oltre agli elementi principali sono presenti i sottoelementi:

- *operation*: descrivono le operazioni
- *message*: descrivono i messaggi
- *endpoint*: descrivono gli endpoint del WS

4.4.2 Description

Description è la radice del documento WSDL, al suo interno sono presenti tutti gli elementi specificati dallo standard. L'elemento descripion ha:

- Un nome locale descripion
- Deve essere qualificato con namespace *http://www.w3.org/ns/wsdl*
- Deve avere un attributo *targetNamespace*
- Può dichiarare zero o più namespace
- Può dichiarare zero o più elementi di informazione documentation
- Può contenere l'elemento types
- Può contenere zero o più elementi interface
- Può contenere zero o più elementi binding
- Può contenere zero o più elementi service

Vediamone un esempio:

< ?xml version="1.0" encoding="UTF-8"?>
< description xmlns="http://www.w3.org/ns/wsdl"
targetNamespace="http://www.esempio.com/tamagninistella" xmlns :
pre="http://www.esempio.com/tamagninistella < /description>

4.4.3 Documentation

L'elemento documentation contiene la descrizione del servizio in formato testuale, il motivo della sua realizzazione, il significato di ogni messaggio scambiato tra servizio e fruitore, la sequenza ed il modo in cui le operazioni devono essere invocate. Tali informazioni sono leggibili da un essere umano e sono racchiuse tra i due tag *<documentation>* e *</documentation>*.

4.4.4 Types

L'elemento types è un elemento opzionale della radice description che contiene la definizione dei tipi di dato presenti nei messaggi di ingresso e d'uscita scambiati con il WS. Tale definizione può essere realizzata mediante l'uso di un qualsiasi linguaggio di definizione di grammatiche, anche se, per ottenere la massima interoperabilità e neutralità dalla piattaforma è consigliato l'uso di XML schema. I tipi di dati utilizzati nei messaggi scambiati tra il richiedente e il fornitore del servizio devono essere dichiarati come singoli elementi e possono caratterizzare tipi semplici o strutture dati complesse.

4.4.5 Interface

L'elemento Interface rappresenta l'interfaccia astratta (resa successivamente concreta grazie al binding) di un WS contenente la definizione di un serie di operazioni astratte che descrivono l'interazione tra fruitore e fornitore del servizio. Ogni operazione specifica il tipo dei messaggi che possono essere inviati o ricevuti dal servizio. Ogni elemento interface ha:

- Un nome locale interface
- Deve essere qualificato con namespace
- Un attributo name che identifica in modo univoco un'interfaccia
- Può avere un attributo opzionale extends che permette all'interfaccia di ereditare le operazioni definite in una o più interfacce specificate in altri documenti WSDL. In questo modo l'interfaccia possiede tutte le operazioni ereditate dall'interfaccia che estende più quelle direttamente definite da essa
- Può avere un attributo opzionale style *Default* utilizzato per definire un valore di default per gli attributi style di tutte le operazioni previste dalla presente interfaccia
- Può contenere zero o più attributi, qualificati attraverso l'utilizzo dei namespace
- Può contenere zero o più elementi fault

- Può contenere zero o più elementi operation
- fault per specificare il formato del messaggio per ogni errore

Vediamo a sua volta la composizione dell'elemento operation:

- Nome locale operation
- Deve essere qualificato
- Un attributo name
- Può contenere l'attributo pattern, il cui valore è un URI che identifica il MEP (Message Exchange Pattern) utilizzato dall'operazione
- Può contenere l'attributo style, il cui valore è un URI che identifica un insieme di regole rispettate nel definire l'operazione
- Può contenere l'attributo safe appartenente al namespace "*http://www.w3.org/ns/wsdl-extensions*"
- Può contenere zero o più elementi **input** che specificano il formato del messaggio di input in ingresso all'operazione considerata. L'elemento input possiede gli attributi *messageLabel* ed *element* che indicano rispettivamente il ruolo che il messaggio ha nel MEP dell'operazione e il tipo di dati utilizzato dal messaggio.
- Può contenere zero o più elementi *output*, ha lo stesso significato dell'elemento input ma applicato al messaggio di output dell'operazione.
- Può contenere zero o più elementi *outfaul*t e/o *infault* che specificano il formato del messaggio di fault in uscita o ingresso all'operazione considerata. Tali elementi possiedono gli attributi *ref* e *messageLabel.* L'attributo *ref* riferisce un fault precedentemente definito all'interno di interface attraverso il suo nome. L'attributo *messageLabel* indica il ruolo che il fault ha nel MEP dell'operazione.

4.4.6 Binding

Lo scopo dell'elemento binding è quello di definire un formato concreto per i messaggi scambiati tra cliente e servizio e un protocollo di

trasporto per ogni operazione e fault presenti nell'interfaccia definita nella sezione precedente. E' possibile che all'interno del medesimo file WSDL la stessa interface, e quindi il servizio, possa essere accessibile, da diversi protocolli di trasporto grazie al binding. L'elemento binding contiene due sottoelementi

- **fault**: specifica i dettagli dei fault attraverso due attributi
  - *ref*: identifica un fault definito in precedenza nella sezione interface
  - *code*: specifica il codice di errore per indicare il tipo di fault
- **operation**: specifica i dettagli delle operazioni attraverso due attributi:
  - *ref*: identifica una operazione definita nella sezione interface
  - *mep*: attraverso una URI specifica come implementare il MEP

Di seguito riportiamo un esempio:

```
<binding name="prenotazioneSOAPBinding"
interface="prefix:prenotazioneInterface" type="http://www.w3.org/ns/wsdl/soap"
prefix2:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="prefix:Disponibilita"
    prefix2:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    <fault ref="prefix:DataFault" prefix2:code="soap:Sender"/>
```

4.4.7 Service

Service specifica l'interfaccia del servizio e gli endpoint attraverso i quali il servizio è disponibile. Ogni endpoint si riferisce ad uno specifico binding definito in precedenza (sezione binding) nel quale è stato specificato il protocollo di trasporto e il formato concreto dei messaggi. Il sottoelemento endpoint, è composto dai seguenti attributi:

- **name**: definisce il nome dell'endpoint
- **binding**: specifica il nome del binding (definito nella sezione binding) utilizzato dall'endpoint

- **address**: specifica l'indirizzo fisico attraverso il quale è possibile accedere al servizio utilizzando il binding descritto dall'attributo binding

Vediamo un esempio:
<service name="DisponibilitaService" interface="prefix:prenotazioneInterface">
<endpoint name="prenotazioneEndpoint"
binding="prefix:prenotazioneSOAPBinding" address
="http://esempio.com/prenotazione"/> </service>

4.4.8 Message Exchange Patterns

I MEP presenti nella definizione delle operazioni (operation) all'interno della sezione interface descrivono l'interazione tra cliente e fornitore del servizio a un livello astratto. I MEP indicano l'ordine di trasmissione dei messaggi ed i messaggi di errore da inviare quando il flusso di messaggi viene interrotto da un evento qualsiasi. I MEP forniti dalle specifiche 2.0 sono:

- **In-Only**: un messaggio di richiesta inviato dal client al service. In caso di errore non viene propagato nessun fault
- **Robust In-Only**: un messaggio di richiesta inviato dal client, al verificarsi di un errore viene inviato un messaggio di fault al mittente
- **In-Out**: il più utilizzato, c'è una richiesta inviata dal client e la risposta dal server. In caso di errore viene inviato un fault dal service al client. Questo è stato quello scelto per il progetto
- **In-Optional-Out**: identico ad In-Out tranne che la risposta non è obbligatoria
- **Out-Only**: solo un messaggio inviato dal service al client. In caso di errore non è previsto il messaggio di fault. E' prevista una precedente registrazione al servizio da parte del client
- **Robust Out-Only**: un solo messaggio come l'Out-Only, ma con la differenza che in caso di errore viene inviato un fault dal client al server

- **Out-In**: specifica un modello richiesta-risposta in cui la comunicazione viene iniziata dal service. In caso di errore viene inviato un messaggio di fault dal client al service
- **Out-Optional-In**: analogo al pattern Out-In con la differenza che in questo caso la risposta è opzionale. In caso di errore è presente il messaggio di fault

4.5 SSL

Oltre ai protocolli dello stack dei Web services ho adottato anche il Secure Socket Layer per garantire la sicurezza delle operazioni. SSL è un protocollo progettato per fornire la cifratura e l'autenticazione tra un client ed un server web. SSL viene implementato come un nuovo metodo di accesso all'URL (https://...) al quale è stata assegnata la porta 443. Ciò significa che lo stesso server può trattare sia un server HTTP insicuro (sulla porta 80) che un server HTTPS sicuro. SSL è un protocollo a livelli che deve interfacciarsi a livello più basso con un livello di trasporto affidabile come il TCP ed a livello superiore con un protocollo applicazione. Il protocollo SSL fornisce le seguenti caratteristiche:

- **Autenticazione del server SSL**, in modo che un utente possa confermare l'identità del server. Questo è reso possibile dal fatto che un browser SSL-compatibile o un'applicazione standalone mantiene un elenco dove sono memorizzate le *Certification-Authority* (CA) fidate, con le relative chiavi pubbliche. Il browser ottiene dal server SSL un certificato contenente la sua chiave pubblica; tale certificato è firmato proprio da una CA che compare nell'elenco delle CA fidate. In questo modo il browser può verificare l'identità del server, prima che l'utente inoltri dati sensibili
- **Autenticazione del client SSL**, in modo che il server possa autenticare il client. Sebbene questa caratteristica di SSL sia solo opzionale, è molto importante specialmente in casi in cui il server

è una banca che vuole autenticare il cliente prima di mandargli informazioni professionali. L'autenticazione del client avviene in modo analogo a quella del server: anche il client infatti può utilizzare dei certificati rilasciati dalle Certification Authority

- **Sessione SSL cifrata**: le informazioni inviate fra il browser ed il server vengono cifrate in fase di spedizione e decifrate in fase di ricezione. Inoltre SSL fornisce un meccanismo per rilevare eventuali contraffazioni delle informazioni da parte di un intruso

Il protocollo inizia con una fase di handshake tra il client ed il server SSL, durante la quale vengono negoziati i parametri che verranno usati durante la comunicazione (*Negotiation of Crypto-options*). Il client deve infatti accordarsi col server su un algoritmo di cifratura, un algoritmo di integrità e un algoritmo per la cifratura a chiave privata; in caso contrario non viene garantita la sicurezza del canale di comunicazione la richiesta verrà scartata.
La negoziazione delle Crypto-options avviene automaticamente, senza che l'utente debba intervenire. A questo punto può iniziare la trasmissione dei dati, e questi sono cifrati con la chiave concordata durante la fase di handshake.
SSL è una tecnologia ormai largamente utilizzata e compresa in quasi tutti i web server e browser. SSL2 supporta solo l'autenticazione del server, SSL3 supporta anche l'autenticazione del client.

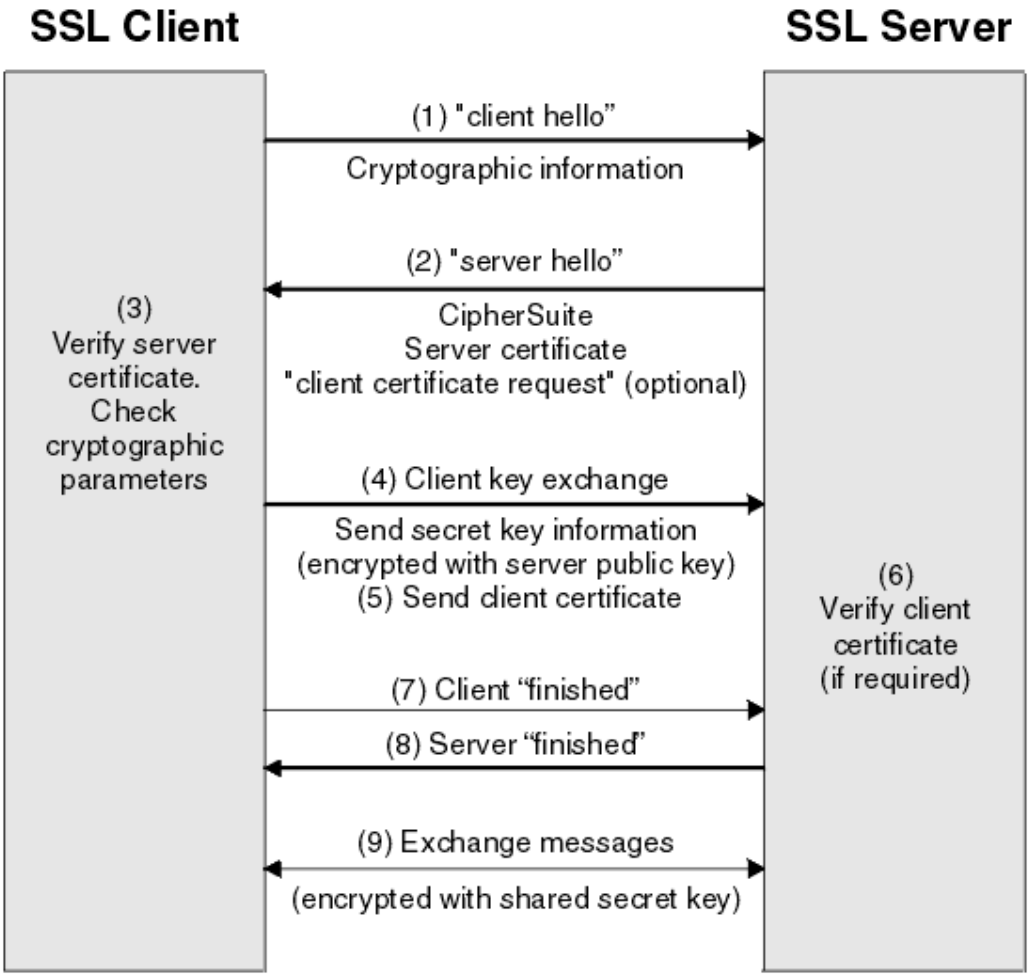

Figura 2: Negotiation of crypto options

4.6 METRO

I web services sono implementati usando la tecnologia Java e dai tool integrati nel suo Web Service Stack chiamato Metro. Lo stack Metro, composto da JAX-WS, JAXB e WSIT, permette allo sviluppatore di creare e mettere in uso web services in grado di collaborare con i client .NET in maniera sicura, affidabile e attraverso l'uso di transazioni. Metro fa parte del progetto Glasshfish e può essere visto diviso in due parti:

- **Core Web Service**: include le tecnologie per l'invocazione ed il deploy dei WS, per la gestione dei documenti XML e la gestione della messaggistica SOAP:
    - Java API for XML-Based Web Services (JAX-WS)

    - Java Architecture for XML Binding (JAXB)
- **Enanched Web Services**: tecnologie WS-based per la gestione di caratteristiche speciali quali consegna affidabile dei messaggi, gestione di transazioni o interoperabilità:
    - Web Services Interoperability Technologies (WSIT): permette la cooperazione tra Java Web Services e Microsoft's Windows Comunication Foundation

4.6.1 JAXB

Java Architecture for XML Binding fornisce API e tools che automatizzano il mapping tra documenti XML e oggetti Java. JAXB facilita l'uso di XML attraverso la compilazione di XML schema in una o più classi Java permettendo di eseguire sui documenti le seguenti operazioni:

- Unmarshall i contenuti XML in "rappresentazioni" java
- Accesso, update e validazione di "rappresentazioni" java rispetto a XML schema
- Marshall di "rappresentazioni" java in contenuti XML

Le annotazioni più importanti, definite da JAXB, sono contenute nel package *javax.bind.annotation.*
Riportiamo in tabella il mapping di default tra tipi di dati XML e tipi Java ottenuti tramite JAXB.

XML schema type	Java data type	XML schema type	Java data type
xs:string	java.lang.String	*xs:boolean*	boolean
xs:integer	java.math.BigInteger	*xs:byte*	byte
xs:int	int	*xs:Qname* *xs:NOTATION*	jax.xml. namespace.QName
xs:long	long	*xs:base64Binary*	byte[]
xs:short	short	*xs:unsignedInt*	long
xs:decimal	java.math.BigDecimal	*xs:time,xs:date,* *xs:dateTime,* *xs:gDay,* *xs:gMonth,* *xs:gMonthDay,* *xs:gYear,* *xs:gYear-Month*	javax.xml.datetype. XMLGregorianCalendar
xs:float	float	*xs:anySimpleType*	java.lang.String
xs:double	double	*xs*	

Tabella 4: binding tra XML schema type e Java data type

4.6.2 JAX-WS

Java API for XML Web Services è una tecnologia che permette di lavorare in maniera facile e potente con i WS. JAX-WS (rinominato così dalla versione 2.0, prima era JAX-RPC) permette di creare e di utilizzare i Web Service mediante l'impiego della tecnologia Java senza richiedere al programmatore una conoscenza profonda di SOAP, XML e di WSDL. Attraverso gli oggetti ed i metodi messi a disposizione da JAX-WS lo sviluppatore può creare messaggi SOAP o invocare i web service ed ottenere le risposte come se stesse lavorando con delle comuni classi Java questo perché gli strati sottostanti sono nascosti. A partire da un WS scritto in Java e l'utilizzo di un web server come Tomcat e di un IDE come Netbeans permette di ottenere in modo automatico il file WSDL, il descrittore di deploy, gli schemi XML, la traduzione tra oggetti Java e messaggi SOAP/XML e viceversa. L'utilizzo delle annotazioni consente di controllare la struttura del file WSDL e di trattare in maniera dichiarativa alcune caratteristiche del comportamento del WS o dei componenti che lo compongono.

Un web server è composto da due componenti:

- **SEI** (Service Endpoint Interface): una interfaccia con i metodi
- **SIB** (Service Implementation Bean): la sua implementazione

L'implementazione può essere realizzata come:

- **Plain Old Java Object** (POJO): semplice classe Java, il metodo utilizzato
- **Session EJB** (Enterprise JavaBean):

Di seguito elenco le notazione più importanti:

- ***@WebService:*** definisce la classe come interfaccia di un web service o come sua classe implementatrice

Proprietà	Descrizione
name	specifica il nome di wsdl:portType
targetNamespace	indica il namespace del servizio
serviceName	specifica il nome di wsdl:service
endpointInterface	usato nel SIB specifica l'implementazione a quale interfaccia fa riferimento
portName	indica il nome di wsdl:port
wsdlLocation	non usata nel JAX-WS 2.0

Tabella 5: proprietà per @WebService

- ***@WebMethod:*** espone un metodo ad essere operazione del web server

Proprietà	Descrizione
operationName	definisce wsdl:operation
action	
exclude	esclude il metodo considerato dalle lista delle operazioni esportate

Tabella 6: proprietà per @WebMethod

- ***@WebParam:*** specifica il mapping tra i parametri della funzione considerata ed i relativi elementi del file WSDL

Proprietà	Descrizione
name	specifica il nome del parametro
targetNamespace	specifica il namespace per quel parametro
mode	specifica la direzione in cui va il parametro
header	specifica se deve essere trasportato nell'header
partName	specifica wsdl:part

Tabella 7: proprietà per @WebParam

- ***@WebResult:*** specifica i dettagli relativi al valore di ritorno restituito dal WS invocato

Proprietà	Descrizione
name	specifica il nome del valore di ritorno
targetNamespace	specifica il namespace relativo al valore di ritorno
header	specifica se il risultato deve essere trasportato nell'header
partName	specifica wsdl:part

Tabella 8: proprietà per @WebResult

- ***@SOAPBinding***: specifica i dettagli del protocollo SOAP per l'invocazione del WS

Proprietà	Descrizione
style	style usato: DOCUMENT o RPC
use	codifica usata; per JAX-WS 2.0 solo LITERAL
parameterStyle	specifica sei parametri sono WRAPPED o BARE??

Tabella 9: proprietà per @SOAPBinding

- ***@WebServiceRef:*** riferisce un WS da una classe Java

Proprietà	Descrizione
name	il nome JNDI della risorsa
type	il tipo java della risorsa
mappedName	il nome in cui la risorsa deve essere mappata
value	la classe del servizio
wsdlLocation	posizione del file WSDL che descrive il servizio

Tabella 10: proprietà per @WebServiceRef

- ***@HttpSessionScope***: indica al container di creare una istanza della classe per ciascuna sessione HTTP

L'utilizzo del framework JAX-WS consente inoltre di generare, a partire dal documento WSDL, le classi necessarie all'invocazione del servizio:

- ❖ da shell tramite il comando *wsimport*
- ❖ da un IDE di sviluppo come Netbeans che automatizza l'import

Il comando *wsimport* effettua il parsing del file WSDL crea le classi stub per invocare il servizio, le classi necessarie alla comunicazione e le classi di binding per gli XML schema specificati. A questo punto il servizio può essere utilizzato come fosse un tipico oggetto java.

4.7 Tomcat

Anche se Internet si è diffusa presso il grande pubblico da poco più di una decina di anni, esiste ormai da oltre trent'anni e in linea generale ha sempre funzionato secondo lo stesso criterio: in pratica si basa sulla circolazione di informazioni tra terminali attraverso il protocollo HTTP secondo il classico schema:

- Richiesta da parte dell'utente tramite client (input)
- Risposta da parte del server (output)

Per soddisfare le richieste dagli utenti vengono messi a disposizione contenuti "statici", niente di molto differente dalle comuni pagine di un libro. Il software che gira sul computer server e che risponde alle chiamate del client e visualizza le pagine richieste è il **Web server.** Il Web server fornisce solamente gli strumenti per accettare le richieste ed inviare contenuti memorizzati su file (pagine HTML, immagini, suoni ecc..). Per realizzare invece applicazioni più complesse dove è prevista un'interattività molto spinta con l'utente, ad esempio portali, siti di e-commerce, community, occorre uno strumento più evoluto del Web server, l'**application server**. L'application server permette di rispondere alle richieste dell'utente tramite veri e propri programmi che vengono eseguiti sul server che danno la possibilità di calcolare la risposta in tempo reale. Esistono numerose tecnologie che permettono l'implementazione di Web server ed application server, una delle più famose è **J2EE** che si avvale dei servizi di un Web server e un application server altrettanto famosi e diffusi: Apache e Tomcat.

Quando un Web server deve rispondere alle richieste del client esegue le seguenti operazioni:

1) Riconosce la richiesta
2) Cerca e, se presente, trova la pagina richiesta
3) Invia la pagina al browser client che la visualizzerà

Il Web server, quindi, **non ha una logica di controllo**, che permetta di interpretare del codice e quindi è capace di gestire contenuti puramente statici, ma non è in grado di garantire alcuna interattività con il visitatore tranne quella del saltare da una pagina ad un'altra tramite i link ipertestuali. Se volessimo interagire con l'utente, permettere di fargli delle domande ed ottenere delle risposte, magari memorizzarle e conservarle per la prossima volta, avremmo bisogno di inserire della logica. Tutto questo è permesso dagli application server. E' impossibile distinguere tra pagine generate da un server "statico" e da una web application, la differenza sta nell'origine di quelle pagine. Le seconde vengono costruite dinamicamente al tempo della chiamata da un programma che gira sul server e che può essere scritto in qualsiasi linguaggio.
Un'applicazione Web, nella maggior parte dei casi, si sviluppa su tre livelli logico-funzionali (**applicazioni Three-Tier**) ma che possono essere distribuiti anche su più livelli (**applicazioni Multi-Tier**):

1. **Livello di presentazione**: rappresenta l'interfaccia utente dell'applicazione e si occupa di acquisire dati e visualizzare risultati
2. **Livello intermedio**: si occupa delle elaborazioni dei dati in base alla cosiddetta business logic, cioè all'insieme delle regole per cui i dati sono considerati significativi e le loro relazioni consistenti; le elaborazioni del livello intermedio generano i risultati richiesti dall'utente
3. **Livello dati**: rappresenta l'insieme dei servizi offerti da applicazioni indipendenti dal Web, come ad esempio un gestore di database, un sistema di gestione di posta elettronica, ecc..

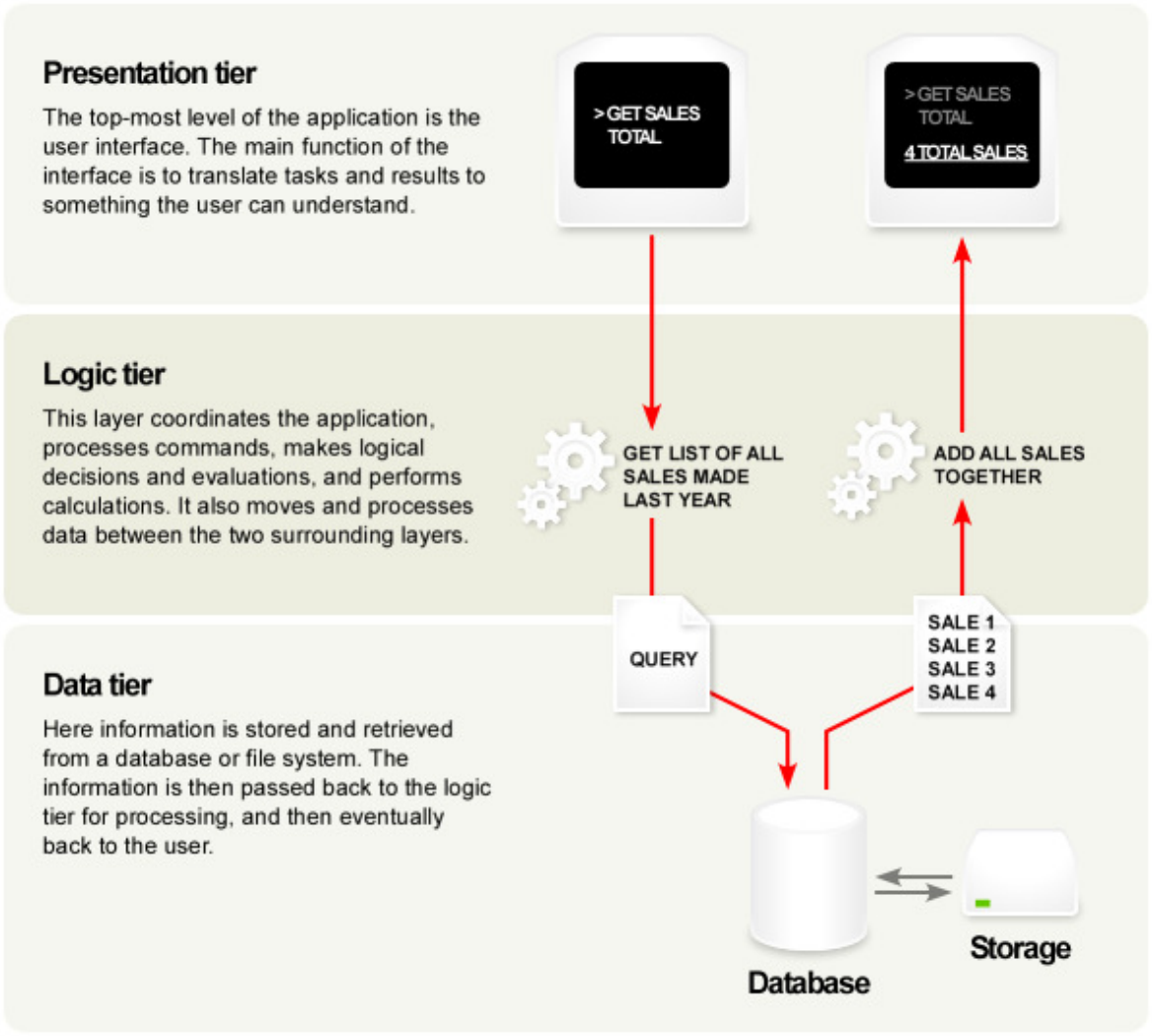

Figura 3: architettura Three Tier

L'esecuzione di questi programmi necessita della presenza sul server di un "ambiente", di un framework per il supporto dei linguaggi che si vogliono utilizzare per la stesura dell'applicazione lato server. L'ambiente che ci dà questo supporto è proprio l'application server. Questo a differenza del Web server, non si limita a rispondere alla richiesta dell'utente con la pagina HTML, ma permette di:

- Affiancare alle normali pagine HTML statiche delle pagine dinamiche
- Inserire lato server dei veri e propri algoritmi

Come abbiamo detto esiste un filo doppio che unisce il linguaggio con cui vogliamo realizzare le pagine dinamiche del nostro sito Web e la sua logica di controllo e l'application server da utilizzare.
La scelta per il linguaggio che vorremo utilizzare per scrivere pagine Web ed applicazioni ricade tra le due famiglie più diffuse **Java** e **DotNet.**
Le due tecnologie hanno parecchio in comune, questo anche perché il Framework .NET ed i relativi linguaggi di programmazione sono stati sviluppati molto dopo la nascita di Java e se ne avverte l'influenza positiva. Ma se è vero che ci sono molti punti in comune, è altrettanto vero che si tratta di piattaforme ben distinte. Scegliere una piattaforma infatti non significa solo scegliere un linguaggio di programmazione, ma anche tutto ciò che lo circonda. Chi usa la tecnologia Microsoft .NET e C#, ad esempio, avrà a che fare con il Framework .NET, con il Web server IIS e con tutte le estensioni (o filtri) ISAPI che gli permettono di svolgere le funzioni di application server.
Chi invece predilige J2EE ha una gamma più larga di scelte. Infatti esistono versioni della JVM sia per Windows sia per Linux ed anche il principe degli AS, Jakarta Tomcat, è disponibile per entrambi i sistemi operativi. Inoltre possiamo scegliere di usare Tomcat con IIS o Apache su sistemi Microsoft o su Apache per sistemi Linux. Questo application server open source è usato per quasi il 70% dei server della rete mondiale ed è la soluzione ottimale per chi sceglie di utilizzare Java e JSP per realizzare il codice delle pagine Web e della logica. Il punto forte della coppia Apache-Tomcat sta nel fatto che, essendo open source, vantano sul supporto delle migliaia di comunità di sviluppatori on line sparsi nei cinque continenti. Per tutte queste ragioni è stato scelto di utilizzare J2EE e Tomcat.
Oltre a quelli già citati vi sono tanti altri AS, tra questi nominiamo:

- **JBoss**
- **WebSphere**
- **Oracle**

Per lo sviluppo di applicazioni Web molto semplici è inutile utilizzare sia Apache che Tomcat visto che quest'ultimo si comporta benissimo anche da Web server. Ma perché proprio Tomcat? Ho scelto Tomcat perché è **semplice da utilizzare, molto stabile ed affidabile, è gratuito e si basa su Java**, la tecnologia Web più diffusa al mondo.
Come per altre tipologie di software, la progettazione e la gestione delle applicazioni Web è resa più agevole seguendo dei paradigmi predefiniti. Il più diffuso è il pattern **MVG** (Model View Controller) che divide l'intero progetto in tre livelli:

1. Il **Modello**, ovvero cosa vogliamo rappresentare e che cosa può essere modificata dall'utente
2. Le **Viste**, ovvero in che modo vogliamo rappresentare il modello, o di parte di esso
3. Il **Controllo**, ovvero la possibilità di interazione con i dati

Appare chiaro come sia quest'ultimo componente quello più significativo nello sviluppo di una Web application. Il **Controller** è il motore dell'applicazione e l'averlo separato dalla visualizzazione dei dati e dal modello ha reso la sua progettazione più semplice permettendo di concentrare gli sforzi sulla logica di funzionamento.
Per l'installazione e l'uso si rimanda ai riferimenti e la sitografia.

4.8 MySQL

Un **Data Base** (traducibile in italiano come "base di dati") non è altro che un insieme di dati logicamente correlati fra loro.
I **Data Base Management System** (DBMS) sono quindi i prodotti software in grado di gestire i database; le loro caratteristiche sono:

- Capacità di gestire grandi quantità di dati
- Condivisione dei dati fra più utenti e applicazioni
- Utilizzo di sistemi di protezione e autorizzazione per l'accesso ai dati stessi

Possiamo identificare diversi tipi di database, in base alla loro struttura logica:

- **Database gerarchici**: basato su strutture ad albero nelle quali ogni dato che non sia a livello radice ha uno o ed un solo padre
- **Database reticolari**: deriva da quello gerarchico, ma rispetto al quale supera la rigidità della struttura ad albero nell'interdipendenza dei dati
- **Database relazionali**: organizza i dati in tabelle, basandosi sulle relazioni fra essi
- **Database a oggetti**: estende i concetti di quello di tipo relazionale adattandoli alla programmazione ad oggetti

I database di tipo relazionale sono, attualmente, di gran lunga i più diffusi ed è il tipo scelto nel progetto.
Il modello relazionale è stato introdotto nel 1970 da Edgar F. Cood col documento "*A Relational Model of Data for Large Shared Data*". Questo modello si basa sulle relazioni fra i dati, i quali vengono presentati in forma tabulare, cioè come un insieme di tabelle ciascuna composta da righe e colonne. Possiamo affermare che:

- Ogni **tabella** contiene i dati relativi ad una entità
- Le **colonne** della tabella rappresentano i campi, ovvero le proprietà o attributi dell'entità
- Le **righe** della tabella esprimono le ricorrenze dell'entità

Insieme al modello relazionale è stato introdotto il **linguaggio SQL** (Structured Query Language), che consente di operare sui dati tramite frasi che contengono parole chiave prese dalla lingua inglese. Del linguaggio SQL sono stati pubblicati vari standard, l'ultimo dei quali risale al 2008 (SQL/2008).

Visto l'ampio successo dei database relazionali sono nati molti **Relational Data Base Management System** (RDBMS) tra i quali citiamo:

- Oracle
- IBM DB2
- Microsoft SQL Server
- PostgreSQL
- MySQL

Ogni software ha le sue caratteristiche, e la grande diffusione dei DB relazionali portò l'inventore del modello a definire, nel 1985, una serie di regole ("le dodici regole di Codd") alle quali un DBMS dovrebbe attenersi per potersi considerare relazionale. Interpretando rigidamente queste regole, sono pochi, forse nessuno, i sistemi che potrebbero propriamente fregiarsi di questo titolo. Anche in MySQL, specialmente nelle versioni non recentissime, mancano alcune funzionalità come le viste e l'integrità referenziale. Tuttavia la versione 5.5 consente a MySQL di essere un vero e proprio DBMS di livello enterprise.
Per l'installazione e l'uso si rimanda ai riferimenti e la sitografia.

5. Il sistema attuale

La realizzazione di questo progetto nasce con l'intento di modernizzare e informatizzare il sistema presente. Attualmente il sistema per la gestione delle ricette mediche è completamente cartaceo:

- Annotazione della ricetta da parte del medico
- Conservazione della ricetta da parte del paziente
- Presentazione della ricetta in farmacia da parte del paziente
- Conservazione della ricetta da parte del farmacista
- Invio, da parte del farmacista, della ricetta per ottenere il rimborso dallo Stato

Ultimamente la Pubblica Amministrazione italiana si sta sensibilizzando sulla necessità di riorganizzare la propria struttura e la modalità del trattamento dei dati. Dopo l'introduzione della Posta Elettronica Certificata l'attenzione del governo italiano si è spostato sulla sanità. Una agenzia ANSA del 14/04/2010 riporta gli annunci fatti dai ministri del governo italiano, in particolar modo, il ministro della Salute e della Funzione Pubblica a tal proposito. Nell'articolo i due ministri annunciano come il futuro che ci aspetta sia fatto da certificati di malattia on-line, ricette mediche via web per controllarne la gestione ed evitarne gli abusi, fascicoli sanitari elettronici e sistemi di centro unico di prenotazione (Cup): in termini tecnici viene definita "dematerializzazione".
Tra le novità, la prima a partire sarà quella dei certificati di malattia via web, che produrrà, secondo i due ministri, risparmi per oltre 500 milioni di euro l'anno, oltre che l'eliminazione di quasi 100 milioni di fogli di carta. Con questa soluzione il medico invierà all'Inps per via telematica i certificati di malattia dei dipendenti pubblici e privati, che non dovranno più inviare tramite raccomandata il loro certificato entro 2 giorni

lavorativi, ma potranno accedere al sistema Inps per visualizzare il proprio attestato, oltre che richiederne al medico copia cartacea o online.
Per quanto riguarda le ricette elettroniche, previste entro la fine dell'anno, consisterà nella possibilità da parte dei medici di poterle spedire via web al ministero dell'Economia, direttamente o tramite il passaggio alle Regioni, con un risparmio stimato intorno ai 5 miliardi di euro l'anno. Un obiettivo è anche quello di avere a disposizione i dati per valutare i casi di appropriatezza, effettuare controlli gestionali e facilitare l'individuazione di truffe o abusi.
Il progetto di tesi propone una implementazione a questo servizio utilizzando la tecnologia dei web services, grazie al quale, oltre alla implementazione della ricetta elettronica si dà supporto al medico durante la compilazione della ricetta e dal lato del farmacista si dà la possibilità di integrare questo strumento con il proprio database del magazzino ad esempio per gestire la sostituzione del medicinale con medicinali simili.
Prima di analizzare la soluzione realizzata illustriamo il sistema delle ricette mediche attualmente in uso.

5.1 La ricetta

Per **ricetta** si intende l'autorizzazione, data in forma scritta al farmacista, perché questi possa consegnare uno o più medicinali al paziente che ne necessita. Una definizione ufficiale di "prescrizione medica" è data dal D.lgs. 24.04.2006 n. 219 (G.U. n. 142 del 21.6.2006 s.o. N. 153 -Codice Comunitario dei medicinali per uso umano - **D.lgs. 219/2006**", in vigore dal 6 luglio 2006). L'art. 1 recita infatti: "*ogni ricetta medica rilasciata da un professionista autorizzato a prescrivere medicinali*". La ricetta ripetibile può essere redatta su un qualsiasi foglio di carta, ma la ricetta non ripetibile deve obbligatoriamente riportare a stampa o mediante un timbro la chiara identificazione del medico proscrittore e dell'eventuale struttura da cui questi dipende art. 89, D.lgs. 219/2006). Le ricette speciali contenenti stupefacenti appartenenti alla Tabella II sezione A e le ricette rilasciate in regime di assistenza SSN

devono essere redatte sugli appositi moduli fissati dalla specifica normativa.
Per "medicinale" si deve intendere in base all'art. 1 D.lgs. 219/2006:

1) Ogni sostanza o associazione di sostanze presentata come avente proprietà curative o profilattiche delle malattie umane
2) Ogni sostanza o associazione di sostanze che può essere utilizzata sull'uomo o somministrata all'uomo allo scopo di ripristinare, correggere o modificare funzioni fisiologiche, esercitando un'azione farmacologica, immunologica o metabolica, ovvero di stabilire una diagnosi medica.

Esistono diversi tipi di ricetta medica, i principali sono:

- Ricetta bianca:
  - Ripetibile
  - Non Ripetibile
- Ricetta rosa: quella utilizzata nell'ambito del SSN
- Ricetta a ricalco

La classificazione dei farmaci segue due codici:

- **CODICE AIC** (Autorizzazione Immissione in Commercio): prima di essere messo in commercio ogni nuovo farmaco viene sottoposto ad una serie di controlli e verifiche da parte del Ministero della Salute. Al termine di tale fase l'AIFA rilascia il relativo codice AIC. Questo codice è alfanumerico ed identifica in modo univoco un solo farmaco
- **CODICE ATC** (Classificazione Anatomica Terapeutica Chimica): questa classificazione è curata dal *Nord Council on Medicine* e dall'*OMS Collaborating Centre for drugs Statistics Methodology* di Uppsala (Svezia). E' costituita da un codice alfanumerico, ha cinque livelli ed è realizzata in funzione dell'organo bersaglio, del meccanismo d'azione e delle caratteristiche chimiche di ogni farmaco

Un'altra classificazione è quella in base al regime di fornitura.

Il D.lgs. 219/2006, art. 87 stabilisce le "*Classi dei farmaci ai fini della fornitura*":

1) Soggetti a prescrizione medica
2) Soggetti a prescrizione medica da rinnovare volta per volta
3) Soggetti a prescrizione medica speciale
4) Soggetti a prescrizione medica limitativa comprendenti:
   a) vendibili su prescrizione di centri ospedalieri o specialisti
   b) utilizzabili in ambiente ospedaliero o ad esso assimilabile
   c) utilizzabili esclusivamente dallo specialista
5) Non soggetti a prescrizione medica:
   a) medicinali da banco o di automedicazione
   b) restanti medicinali non soggetti a prescrizione medica

Gli artt. 88-94 danno "*indicazione obbligatoria del regime di dispensazione*" perché i medicinale devono recare sull'imballaggio esterno o sul confezionamento primario la frase:

a) "Da vendersi dietro presentazione di ricetta medica"
b) "*Da vendersi dietro presentazione di ricetta medica utilizzabile una sola volta*"
c) "Uso riservato agli ospedali. Vietata la vendita al pubblico"
d) Dopo le frasi ci cui al punto a) oppure b) indicare la specificazione del tipo di struttura o specialista autorizzato alla prescrizione
e) "Uso riservato a…" con specificazione dello specialista autorizzato all'impiego del medicinale, e *"Vietata la vendita al pubblico"*

Il **Prontuario Farmaceutico Nazionale** (PFN 2005) indica una *"classificazione dei medicinali ai fini della rimborsabilità"* che per il SSN sono classificati in:

classe a): Farmaci essenziali e per malattie croniche

- gruppo A
- gruppo H:
  - OSP1: utilizzabili solo in ambito ospedaliero
  - OSP2: utilizzabili in ambito ospedaliero o ad esso assimilabile

classe c): Farmaci privi delle sopracitate caratteristiche e non concedibili a carico del SSN

5.1.1 La ricetta "bianca"

La ricetta medica "bianca" è certamente valida se scritta su un comune foglio di carta purché contenga i seguenti elementi essenziali:

1. **Nome** e **cognome del medico** ed eventuale struttura sanitaria di appartenenza
2. **Nome del farmaco** o del **principio attivo**
3. **Luogo** e **data** di compilazione della ricetta
4. **Firma** autografa del medico

Non è necessario indicare il nome e cognome dell'assistito, a meno che il paziente stesso lo richieda. Nella ricetta "bianca" possono essere prescritti un numero indeterminato di farmaci diversi. I farmaci prescritti con questo tipo di ricetta sono sempre a totale carico dell'assistito. Per ottenere farmaci a totale o parziale carico dello Stato, nei casi previsti dalla legge, è indispensabile che il medico utilizzi l'apposito modulo per la prescrizione a carico del SSN. Con la ricetta bianca si possono prescrivere tutti quei farmaci che sulla confezione recano la dicitura: "*Da vendersi dietro presentazione di ricetta medica*". La ricetta "bianca" ha validità non superiore a sei mesi a partire dalla data di compilazione e, comunque, per non più di dieci volte, salvo che per alcune categorie di farmaci (come ormoni o gli ansiolitici), per i quali il

periodo di validità della ricetta è più breve. Entro questi limiti la ricetta è "ripetibile", nel senso che l'assistito può continuare ad esibirla al farmacista per acquistare i farmaci, fino al termine della sua validità. Infatti ogni volta che viene presentata al farmacista per l'acquisto del medicinale, la ricetta viene timbrata ma poi riconsegnata all'assistito per usi futuri. Invece per quei farmaci che per il loro uso continuativo possono determinare stati tossici o, comunque, rischi particolarmente elevati per la salute del paziente, possono essere prescritti soltanto con una ricetta **"non ripetibile"**. In ogni caso questi medicinali recano sulla confezione la dicitura "*Da vendersi dietro presentazione di ricetta medica utilizzabile una sola volta*". Questo tipo di ricetta può essere presentata in farmacia entro trenta giorni dalla data della sua compilazione. Alla presentazione al farmacista, questi, consegna il medicinale e ritira la ricetta.
Il farmaco prescritto può essere sostituito dal farmacista a meno che la ricetta non riporti la clausola **"farmaco non sostituibile"**.

5.1.2 La ricetta del SSN

La ricetta medica a rimborso del SSN deve essere redatta su un apposito modulo, il cosiddetto "ricettario rosa", questo perché deve contenere una formalità e degli elementi in più, rispetto alla "bianca", per far sì che il farmacista si faccia rimborsare dallo Stato il costo dei medicinali forniti agli assistiti. La ricetta del SSN deve contenere gli stessi elementi di quella "bianca" e in più deve avere:

- **Nome** e **cognome** dell'**assistito**
- **Codice fiscale** dell'assistito
- Codice dell'Azienda Sanitaria di riferimento
- **Codici** e motivi di **esenzione**
- **Nota AIFA** (Agenzia Italiana del FArmaco) pertinente

Il farmacista in entrambi i tipi di ricetta può sostituire il farmaco prescritto con un altro a meno che il medico abbia indicato sulla ricetta la dicitura **"farmaco non sostituibile"**. Se questa indicazione non c'è, il

farmacista per legge deve informare l'assistito dell'eventuale esistenza di un farmaco equivalente (cosiddetto "generico") avente il medesimo principio attivo e l'assistito può acconsentire di ricevere il medicinale equivalente al posto di quello di marca. Se però l'assistito si rifiuta di ottenere il medicinale equivalente e pretende comunque il farmaco di marca oppure se quello indicato è non sostituibile, l'assistito è tenuto a pagare la differenza fra il costo del farmaco equivalente (coperto dallo Stato) ed il costo del farmaco di marca.

6. Il Progetto

La realizzazione di questo progetto è nata dall'idea di cambiare il sistema attuale che riguarda la gestione delle ricette mediche per far sì di agevolare il compito di medici, pazienti e farmacisti. Nonostante un sistema informatizzato realizzato per conto del ministero della Salute sia attualmente in fase di sperimentazione, si è voluto comunque creare una alternativa a questa realizzazione, in quanto il valore aggiunto che ne deriva grazie all'implementazione tramite la tecnologia dei web services è enorme. Lo scheletro di questo progetto dà la possibilità di aggiungere facilmente e con il minimo sforzo altri servizi o completare quelli esistenti; in particolar modo mi riferisco all'assistenza che si fornisce a medici e farmacisti nello svolgere il proprio lavoro, automatizzando alcune loro azioni o scelte.
Non si è avuto la pretesa di implementare un sistema già pronto per l'utilizzo reale in quanto per esempio non si sono trattati i casi particolari presenti nella scrittura delle ricette o è mancata una fase accurata di testing. Quello che si è voluto affrontare con questo elaborato di tesi è una prima esplorazione in ampiezza di questo nuovo campo per capirne le dinamiche e scoprire le problematiche.

6.1 Funzionamento generale

Passiamo adesso ad analizzare il funzionamento generale del sistema.
I soggetti che interagiscono tra di loro attraverso la ricetta sono:

- **Medico**
- **Paziente**
- **Farmacista**

Uno dei vincoli di progetto è stato quello di non modificare le interazioni fra questi soggetti per far sì di non creare un impatto troppo violento di spaesamento causato dal cambiamento. Significa quindi che succederanno le seguenti azioni principali:

- Il paziente si reca dal medico per farsi visitare
- Il medico scrive la ricetta in uno dei due tipi
- Il paziente consulta le sue ricette
- Il paziente si reca in farmacia per prendere i farmaci
- Il farmacista controlla la ricetta, dà il farmaco al paziente

Allora dove sono i vantaggi nell'utilizzo della ricetta online?

a. Riduzione dei costi grazie all'eliminazione della carta
b. Il medico ha a disposizione automaticamente ed in qualsiasi momento la storia delle prescrizioni per ogni suo paziente
c. Il paziente non ha più la preoccupazione di conservare la ricetta e portarla con sé in caso ne abbia bisogno
d. Nel caso di ricette riguardanti farmaci per patologie croniche il rinnovo della ricetta può essere effettuato senza che il paziente si rechi dal medico; sarà il sistema ad avvertire il medico di quali ricette sono in scadenza e poi sarà lui a decidere quali rinnovare
e. Il farmacista viene esonerato dal compito di conservare e/o spedire le ricette
f. Sono più difficili le frodi
g. Sono più facili i controlli
h. Si crea una base di dati su cui è possibile fare studi, statistiche e ricerche

Il sistema è composto da:

- ❖ Il server della ASL che a livello regionale fornisce i servizi
- ❖ Un programma installato nel computer di ogni medico di famiglia
- ❖ Un programma installato nel computer di ogni farmacia
- ❖ Un server di magazzino in ogni farmacia
- ❖ Un browser web installato nel computer di ogni paziente

Il server della ASL interagisce con gli altri client e fa da centro stella a questa rete.

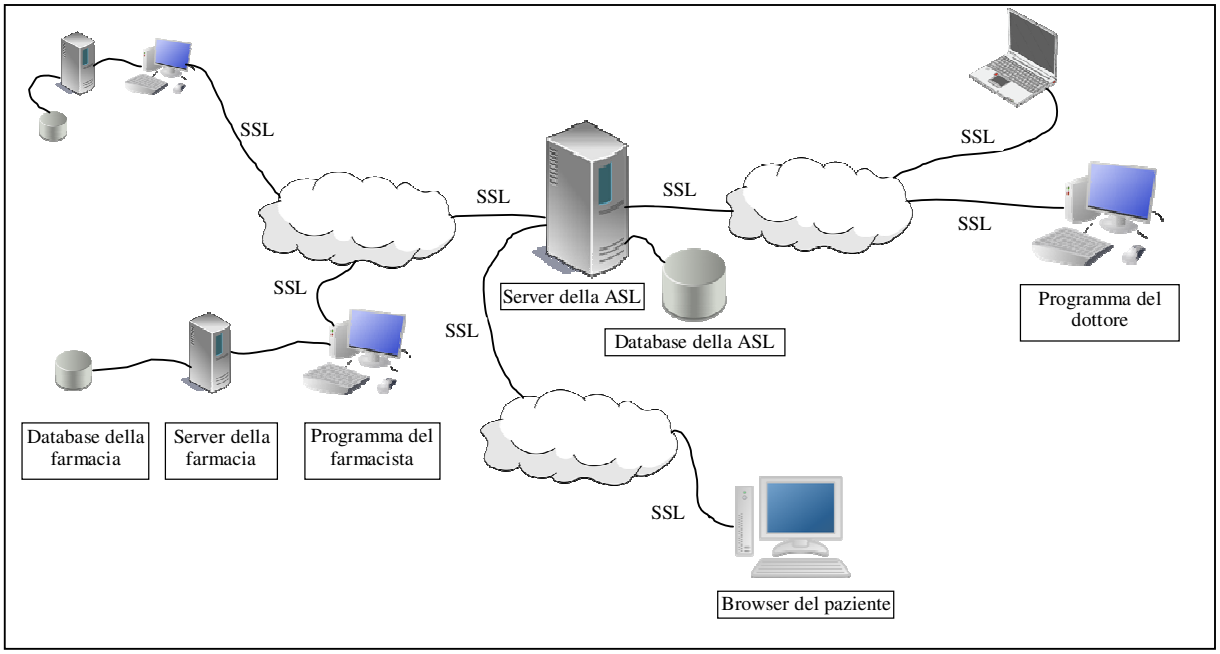

Figura 4: Schema della rete

6.2 Azioni del medico

Il medico avviando il programma si trova davanti alla seguente interfaccia grafica.

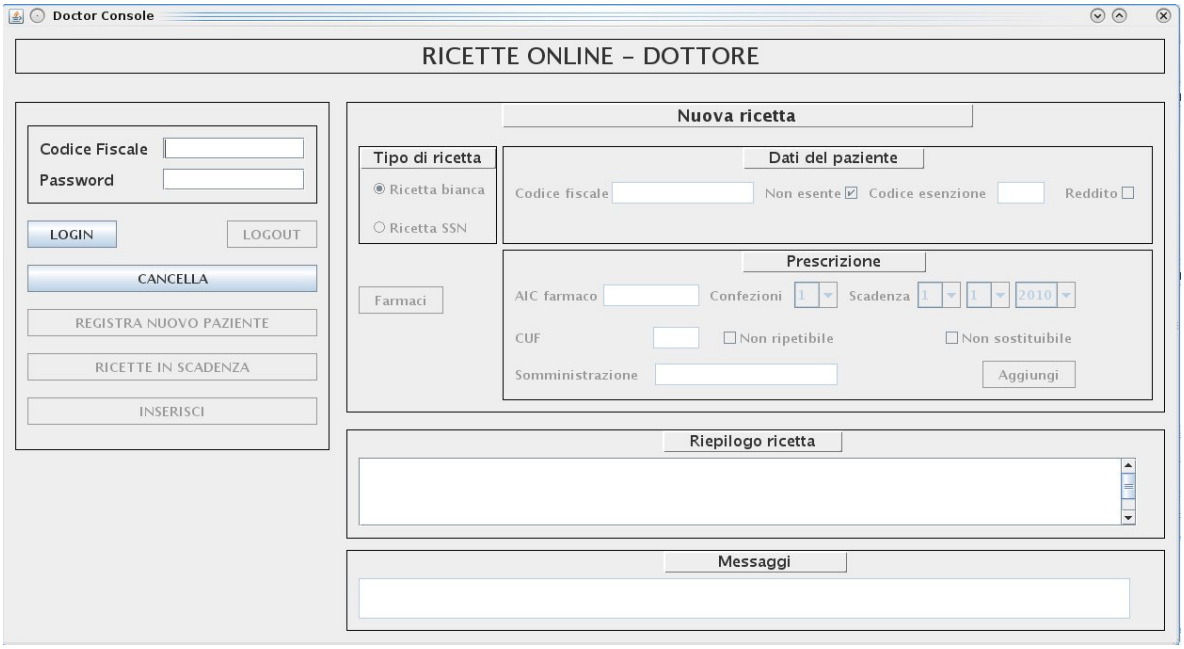

Figura 5: Interfaccia grafica del dottore

Sul lato sinistro sono raggruppati i pulsanti del menù principale, alcuni di questi non sono attivi perché si attiveranno solo dopo il login; i pulsanti sono:

- **LOGIN**: il medico accede al sistema fornendo le sue credenziali:
  - Codice Fiscale
  - Password
- **LOGOUT**: il medico esce dal sistema
- **CANCELLA**: cancella i campi del codice fiscale e password
- **REGISTRA NUOVO PAZIENTE**: apre un'altra interfaccia per la registrazione al sistema di nuovi pazienti
- **RICETTE IN SCADENZA**: apre un'altra interfaccia per la visualizzazione e l'eventuale rinnovo delle ricette in scadenza per quel giorno riguardanti tutti i pazienti del medico
- **INSERISCI**: consente di registrare la ricetta una volta che si è finito di compilarla

La console centrale è formata dal form per compilare le ricette intitolato *"Nuova ricetta"*. Anche questa parte è disattivata fino a che non si effettua il login. Alcuni campi saranno attivi per entrambi i tipi di ricette, altri saranno specifici e attivi solo per un tipo. Subito in alto a sinistra

c'è un selettore per scegliere fin da subito il tipo di ricetta. Il form alla sua destra, riguarda i *"Dati del paziente"* e sono:

- **Codice fiscale**
- **Non esente**(SSN): un checkbox per indicare se il paziente è o meno esente per qualche causa
- **Codice di esenzione**(SSN): in caso di esenzione, si attiva per consentire l'inserimento del codice che identifica il tipo di esenzione
- **Esenzione per reddito**(SSN): è un checkbox che si attiva se il paziente è esente e serve per indicare che la sua esenzione è causata dal reddito

Sotto c'è la parte relativa alla *"Prescrizione"*, ovvero che riguarda il tipo di ricetta ed i farmaci che si stanno inserendo; anche in questo campo alcuni campi saranno attivi o meno in base al tipo di ricetta scelta:

- **AIC:** codice AIC del farmaco
- **Confezioni**: numero di confezioni da prendere per quel farmaco
- **Scadenza**: data di scadenza entro la quale poter acquistare quel farmaco
- **Non ripetibile**(Bianca): ripetibilità o meno della ricetta
- **Non sostituibile**: sostituibilità o meno del farmaco

I pulsanti presenti servono per:

- **FARMACO**: lanciare il browser di sistema alla pagina relativa ad informazioni sui farmaci in commercio; questo fa sì di raccogliere informazioni sul farmaco, farmaci simili ed il suo codice AIC.
- **AGGIUNGI**: aggiungere il farmaco alla ricetta

Più in basso c'è un'area *"Riepilogo ricetta"* riservata all'elencazione dei medicinali prescritti nella ricetta e ancora più in basso c'è un'area per visualizzare i *"Messaggi"* del programma; sia per quelli di successo e conferma sia per quelli di errore.

La prima operazione che deve compiere il medico è quella di fornire le proprie credenziali e loggarsi al sistema. A questo punto, se l'accesso è andato a buon fine, può scegliere se:

- Aggiungere un nuovo paziente al sistema premendo **"REGISTRA NUOVO PAZIENTE"**
- Controllare le ricette in scadenza oggi premendo **"RICETTE IN SCADENZA"**
- Prescrivere una nuova ricetta: compilando il form e premendo **"AGGIUNGI"** ogni volta per aggiungere un nuovo farmaco e alla fine premendo **"INSERISCI"**
- Uscire dal sistema premendo **"LOGOUT"**
-

Il controllo che viene fatto lato client è sulla correttezza del pattern del codice fiscale e sulla lunghezza minima della password. Lato server invece sarà controllata la corrispondenza tra codice fiscale e password.

6.2.1 Registrazione di un nuovo paziente

Il dottore una volta esser entrato nel sistema, premendo il tasto **"REGISTA NUOVO PAZIENTE"** avvia una nuova GUI.

Figura 6: Interfaccia per l'inserimento di un nuovo paziente

Sempre sul lato sinistro è presente il menù con i pulsanti per le azioni:

- **REGISTRA**: invia i dati presenti nel form per consentire la registrazione del nuovo paziente
- **CANCELLA**: cancella i dati presenti nel form

Nella zona centrale è presente il form *"Dati del paziente"* dove inserire appunto tutti i dati necessari alla registrazione del paziente:

- **Cognome**
- **Nome**
- **Codice Fiscale**
- **Password**
- **Ripeti password**

Il controllo che si effettua lato client è sulla correttezza del pattern del codice fiscale, sulla lunghezza minima della password scelta e sulla corrispondenza tra la password e la sua ripetizione. Il server invece si accerterà che quel codice fiscale non sia già presente.

6.2.2 Visualizzare le ricette in scadenza

Se il medico vuole visualizzare le ricette che sono in scadenza nella data odierna riguardanti tutti i suoi pazienti non deve far altro che premere il pulsante **"RICETTE IN SCADENZA"** della console principale.

Figura 7: Interfaccia per la visualizzazione delle ricette in scadenza

Dopo aver premuto il pulsante si aprirà, lasciando la precedente in background, una nuova interfaccia con un menù di pulsanti sul lato sinistro:

- **BIANCHE**: elenca le ricette bianche in scadenza
- **SSN**: elenca le ricette del SSN in scadenza

Una volta premuto il pulsante **"BIANCHE"**, nella zona intitolata *"Ricette Bianche"* compare un elenco delle ricette di tipo bianco in scadenza oggi.

Figura 8: Interfaccia che mostra le ricette bianche in scadenza

Ogni riga corrisponde ad un medicinale e per ciascuna riga viene indicato:

- **Codice Fiscale**: codice fiscale del paziente
- **Cognome**
- **Nome**
- **AIC**: AIC del farmaco prescritto
- **Farmaco**: nome del farmaco
- **Cat.**: categoria di appartenenza
- **Quantità**: il numero di confezioni prescritte
- **Ripetibilità**: se la ricetta è ripetibile
- **Sostituibilità**: se il farmaco è sostituibile

- **Scadenza**: la scadenza futura che si intende indicare
- **Somministrazione**: le modalità di somministrazione del farmaco
- **MODIFICA**: tasto per attivare alcuni campi della riga e modificarli al fine di modificare la prescrizione, apportando modifiche per esempio al farmaco, alle dosi o alla scadenza
- **RINNOVA**: tasto per inviare i nuovi dati della ricetta e quindi per rinnovarla

Premendo invece il tasto **"SSN"**, viene compilata la zona intitolata *"Ricette SSN"* con le ricette del SSN in scadenza oggi.

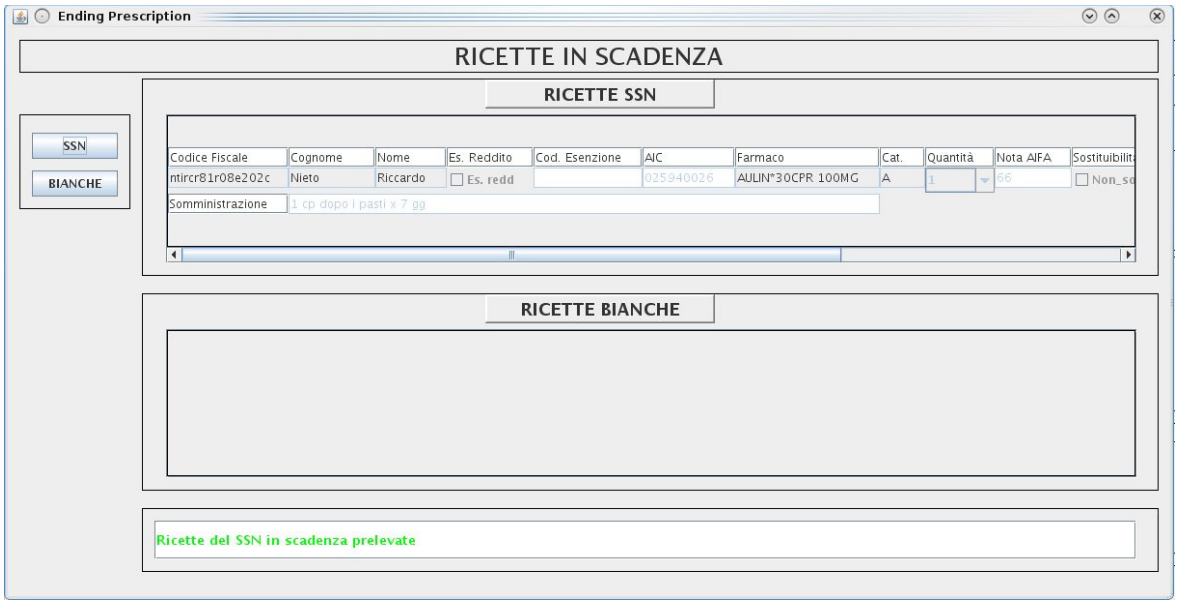

Figura 9: Interfaccia che mostra le ricette del SSN in scadenza

Ogni riga corrisponde ad un medicinale e per ciascuna riga viene indicato:

- **Codice Fiscale**: codice fiscale del paziente
- **Cognome**
- **Nome**
- **Es. Reddito**: indica se il paziente è esente per reddito
- **Cod. Esenzione**: il codice di esenzione, se esente
- **AIC**: AIC del farmaco prescritto
- **Farmaco**: nome del farmaco
- **Cat.**: categoria di appartenenza

- **Quantità**: il numero di confezioni prescritte
- **Nota AIFA**: nota AIFA associata a quel farmaco
- **Sostituibilità**: se il farmaco è sostituibile
- **Somministrazione**: le modalità di somministrazione del farmaco
- **MODIFICA**: tasto per attivare alcuni campi della riga e modificarli al fine di modificare la prescrizione, apportando modifiche per esempio al farmaco, alle dosi o alla scadenza
- **RINNOVA**: tasto per inviare i nuovi dati della ricetta e quindi per rinnovarla

6.2.3 Creazione di una nuova ricetta bianca

Qualora il medico dovesse inserire una nuova ricetta bianca, deve assicurarsi di trovarsi nella console principale e che il pulsante **"Ricetta bianca"** presente all'interno del form sia selezionato. Questa configurazione porterà l'interfaccia in questo stato:

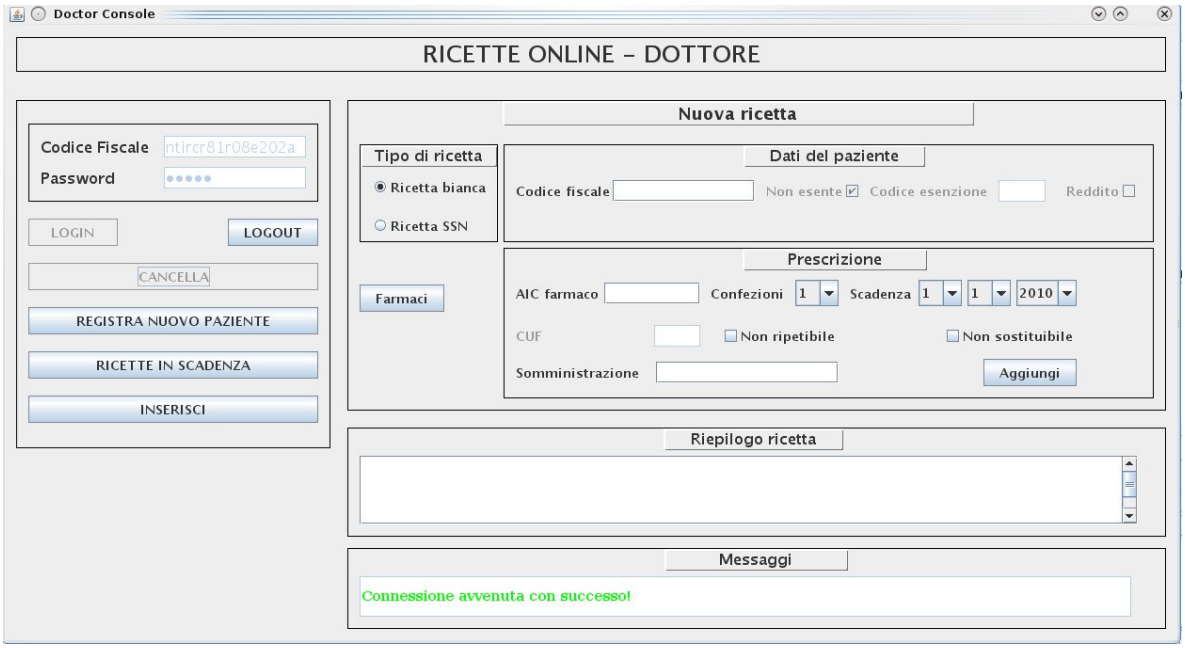

Figura 10: Interfaccia per l'inserimento di una nuova ricetta bianca

Selezionare il JRadioButton per scegliere il tipo di ricetta, causerà l'azzeramento dei campi del form, la pulizia del campo del riepilogo della ricetta e l'attivazione o meno di alcuni campi o selettori.

Il medico una volta compilati i vari campi, specialmente quelli relativi al farmaco, aggiungerà tale farmaco alla ricetta proprio grazie al pulsante **"AGGIUNGI"**; se non ci sono stati errori nella compilazione apparirà un riepilogo della ricetta con i farmaci aggiunti precedentemente più l'ultimo. Il medico eseguirà questa operazione tante volte quanti sono i farmaci che intende aggiungere alla ricetta.

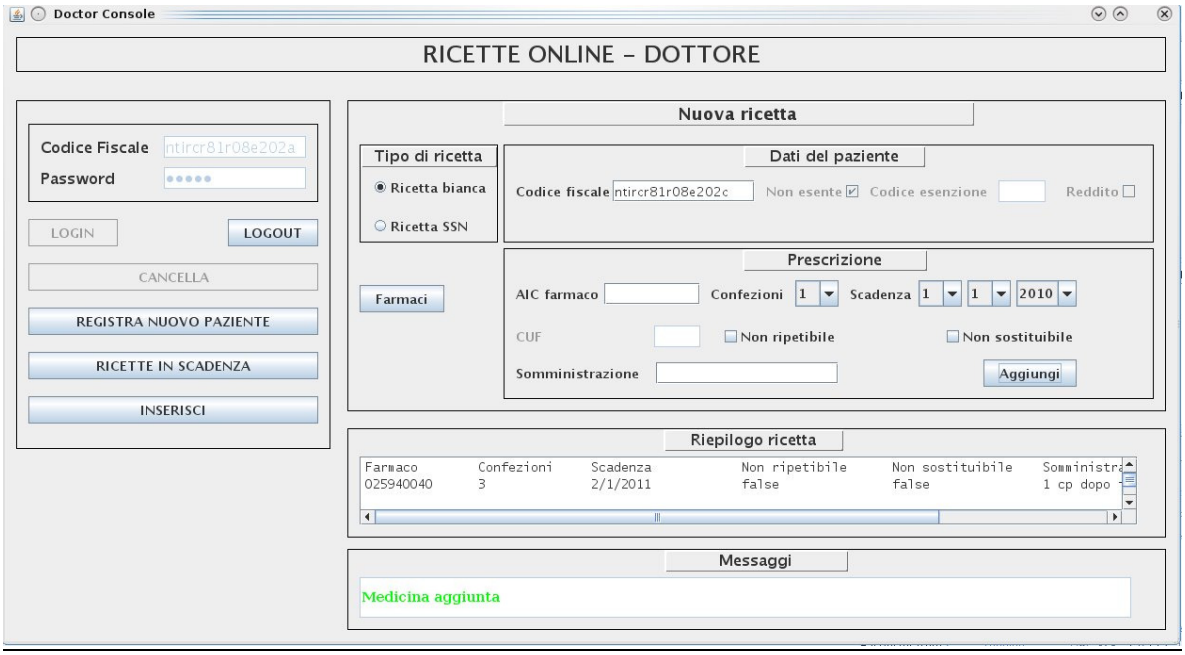

Figura 11: Interfaccia del dottore dopo aver inserito un farmaco nella ricetta bianca

Una volta compilata la ricetta, per inviarla e registrarla nel database dovrà premere il tasto nel menù principale sul lato sinistro "INSERISCI".

Se tutto è andato a buon fine apparirà un messaggio di conferma ed il form sarà ripulito.

Durante la compilazione della ricetta, dopo aver inserito il codice AIC del farmaco ed aver selezionato la sua ripetibilità o meno, il campo scadenza si aggiorna automaticamente con le scadenze massime possibili. Il dottore avrà comunque la possibilità di cambiare questa scadenza, accorciandola. Con la pressione del tasto **"AGGIUNGI"** la scadenza sarà sempre controllata, in modo che abbia un valore valido, compreso tra la data odierna e quella massima possibile in base al tipo di

ricetta che si sta compilando. Lato server sarà controllata l'esistenza del farmaco indicato.

6.2.4 Creazione di una nuova ricetta del SSN

Se il medico dovesse inserire una ricetta a carico del SSN dovrà, invece, assicurarsi di trovarsi nella console principale, ma che sia selezionato il bottone **"Ricetta SSN"**. Questa configurazione porterà l'interfaccia in questo stato.

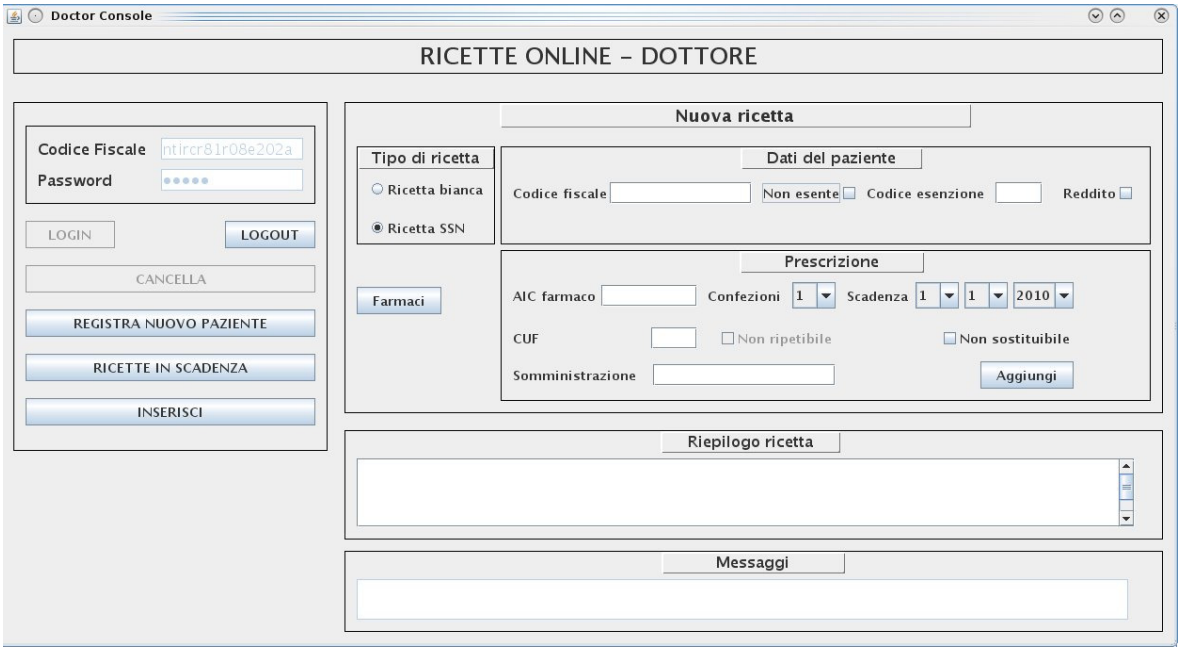

Figura 12: Interfaccia del dottore per inserire una nuova ricetta del SSN

Anche in questo caso la selezione del JRadioButton per scegliere il tipo di ricetta, causerà l'azzeramento dei campi del form, la pulizia del campo del riepilogo della ricetta e l'attivazione o meno di alcuni campi o selettori.

Il medico una volta compilati i vari campi, specialmente quelli relativi al farmaco, aggiungerà tale farmaco alla ricetta proprio grazie al pulsante **"AGGIUNGI"**; se non ci sono stati errori nella compilazione apparirà un riepilogo della ricetta con i farmaci aggiunti precedentemente più l'ultimo. In questo caso l'azione di aggiunta di un nuovo farmaco sarà

consentita solo rispettando i limiti di massimo due farmaci e con un massimo di confezioni pari a tre.

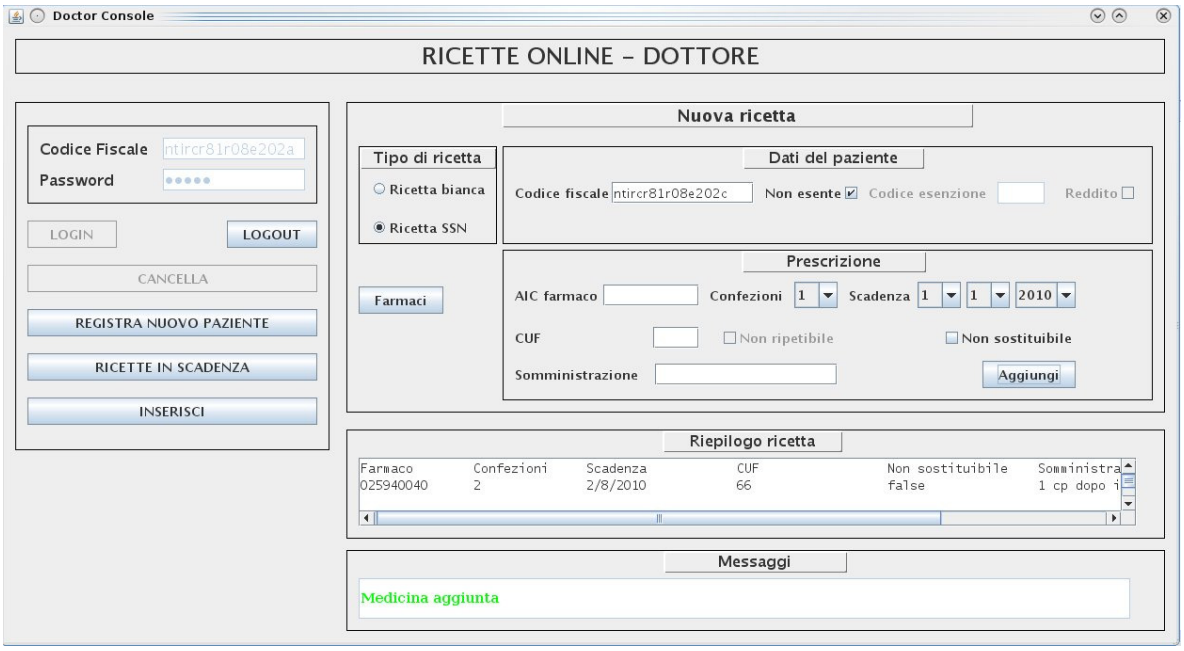

Figura 13: Interfaccia del dottore dopo l'inserimento di un farmaco nella ricetta del SSN

Una volta compilata la ricetta, per inviarla e registrarla nel database dovrà premere il tasto nel menù principale sul lato sinistro **"INSERISCI"**.

Se tutto è andato a buon fine apparirà un messaggio di conferma ed il form sarà ripulito.

Anche in questo caso dopo aver inserito il codice del farmaco il campo per la scadenza si aggiorna automaticamente al valore massimo possibile per quel tipo di ricetta. Successivamente il programma controllerà la validità del campo, cioè se la data indicata è compresa tra la data odierna e quella massima possibile. Lato server c'è il controllo sull'esistenza del farmaco prescritto.

6.3 Azioni del paziente

Il paziente in qualsiasi momento utilizzando un browser può venire a conoscenza delle ricette che gli sono state prescritte. Dopo aver digitato l'indirizzo del servizio ed aver avviato la navigazione; il browser

chiederà all'utente di accettare il certificato che gli è stato mandato. Una volta accettato inizierà la navigazione protetta da SSL e le volte successive non sarà più richiesto questo passaggio perché il certificato sarà già presente in memoria e sarà riconosciuto automaticamente.
Il paziente a questo punto si troverà davanti questo sito.

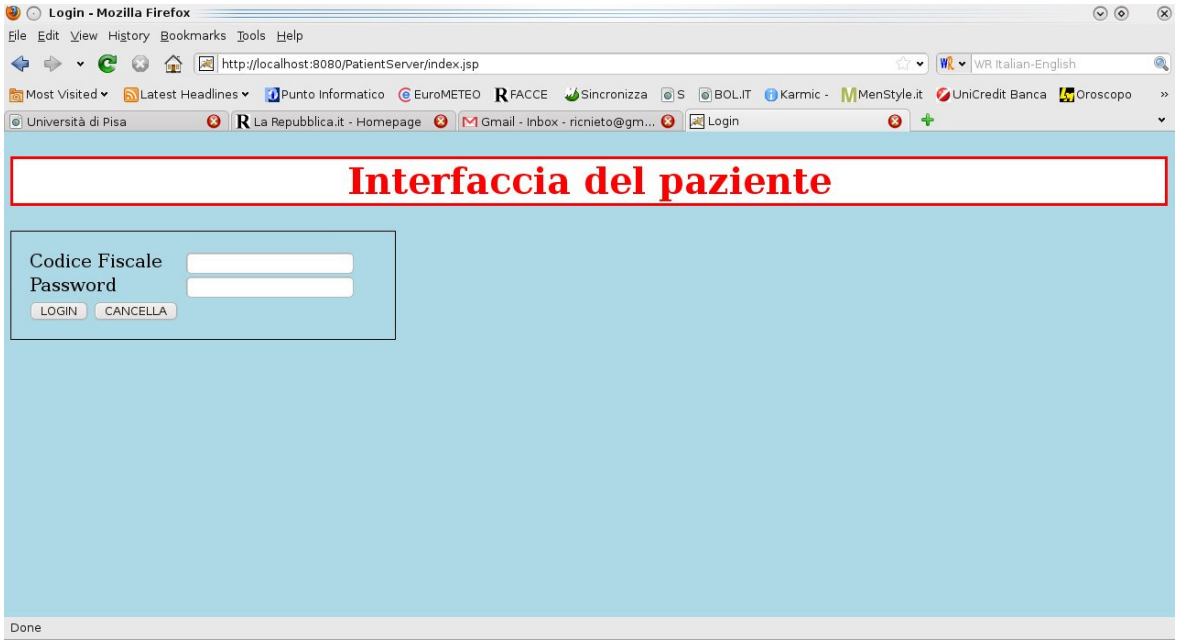

Figura 14: Pagina di login per il paziente

Per adesso l'interfaccia grafica è molto semplificata e le funzioni disponibili sono limitate, ma gli sviluppi futuri sono molteplici.
Questa prima pagina è la pagina di login, sulla sinistra ci sono i campi per inserire codice fiscale e password.
Premendo il tasto **"LOGIN"** l'utente entra nel sistema e accede alle sua pagina di navigazione principale.

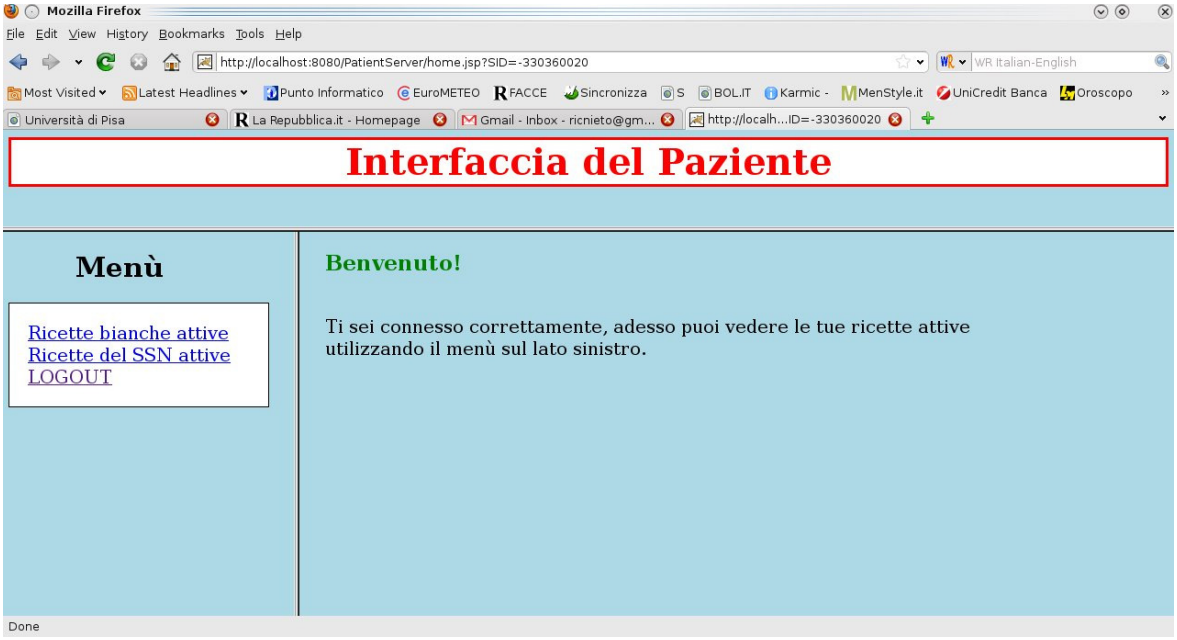

Figura 15: Pagina iniziale per la navigazione

Nella console principale compare un messaggio di benvenuto che invita ad usare il menù nel frame a lato per effettuare le operazioni volute. Il frame centrale sarà il contenitore delle pagine richieste attraverso il menù.

Cliccando sul link ***"Ricette bianche attive"*** il paziente visualizza una tabella in cui sono presenti le informazioni relative alle sue ricette bianche attive:

- **Farmaco**: nome del farmaco prescritto
- **Conf.**: numero di confezioni prescritte
- **Util.**: numero di confezioni già acquistate
- **Scadenza**: scadenza per l'acquisto del farmaco
- **Somministrazione**: modalità di somministrazione del farmaco

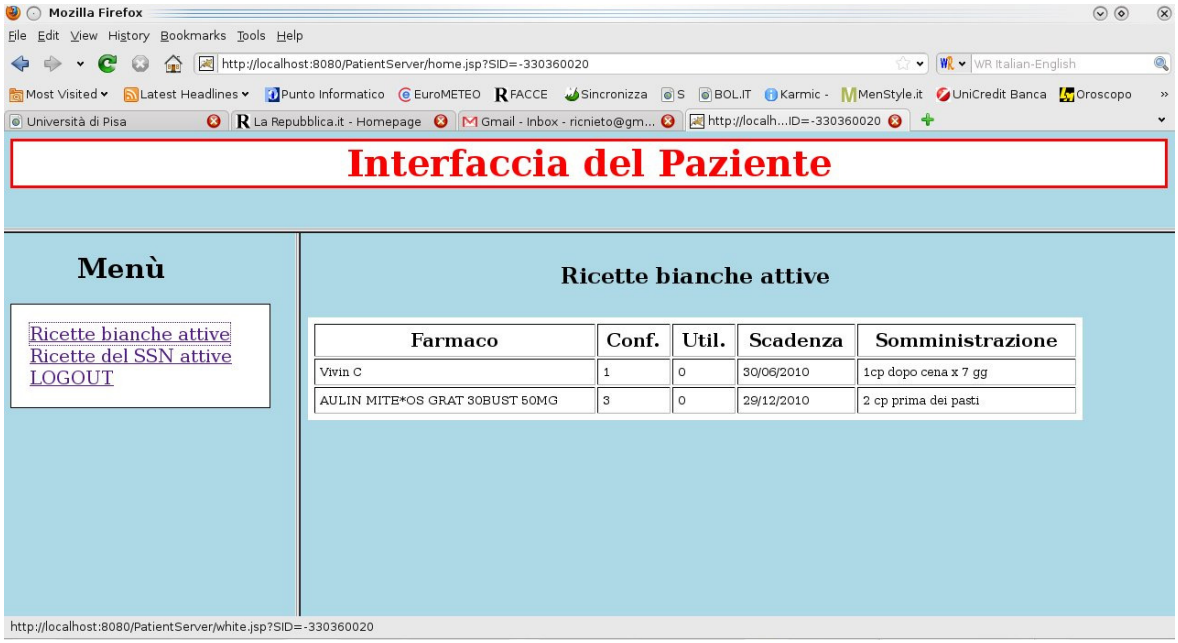

Figura 16: Visualizzazione delle ricette bianche attive

Allo stesso modo, cliccando sul link ***"Ricette del SSN attive"*** visualizza una tabella in cui sono presenti le stesse informazioni, ma relative alle sue ricette del SSN.

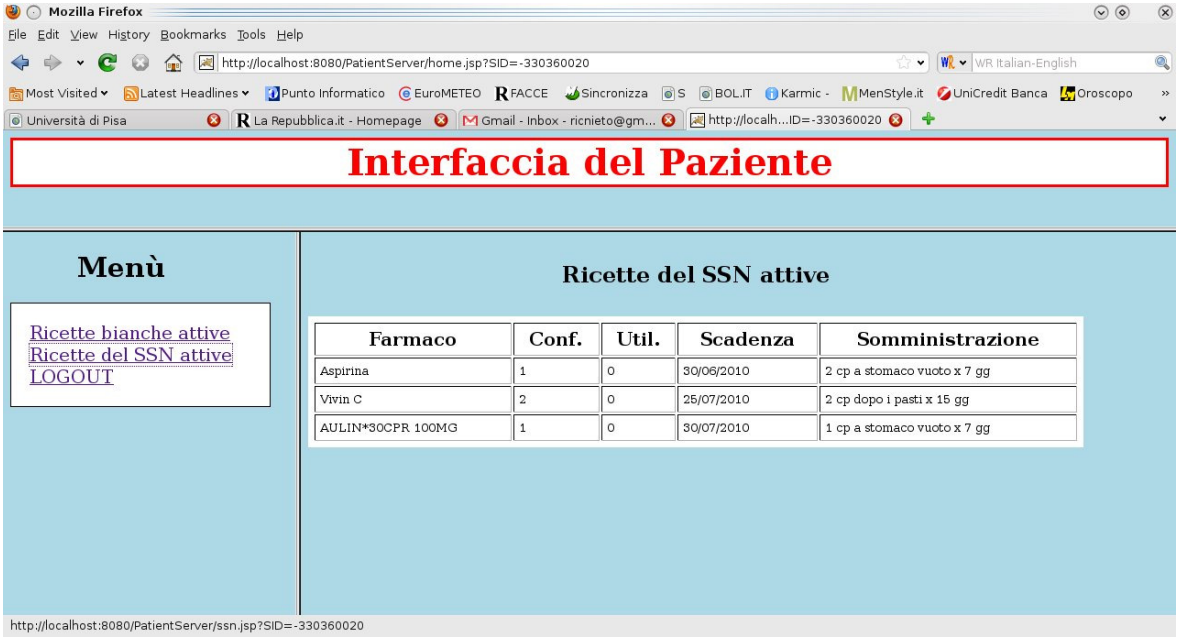

Figura 17: Visualizzazione delle ricette del SSN attive

Una volta terminata la navigazione il paziente esce dal sistema cliccando su ***"LOGOUT"*** e verrà visualizzata la pagina di conferma dell'avvenuta uscita.

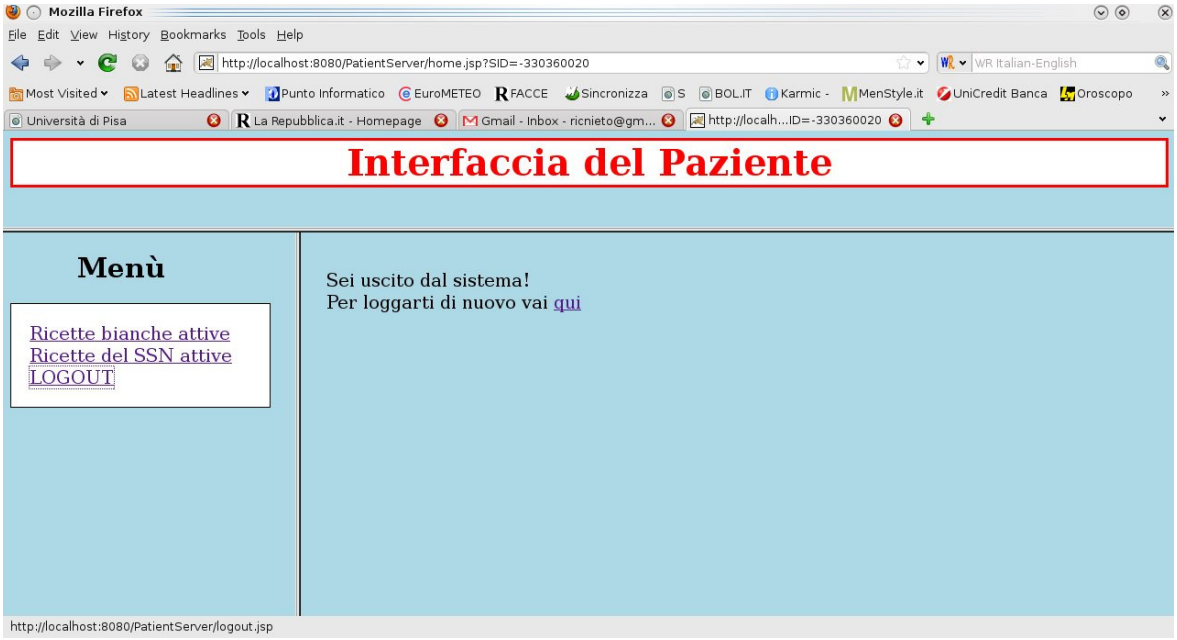

Figura 18: Pagina di conferma del logout

I controlli effettuati in questo caso sono lato client sulla correttezza del pattern per il codice fiscale e la lunghezza minima della password. Lato server sarà controllata la correttezza dell'associazione codice fiscale e password.

6.4 Azioni del farmacista

Il farmacista avviando il suo programma si troverà davanti questa interfaccia grafica.

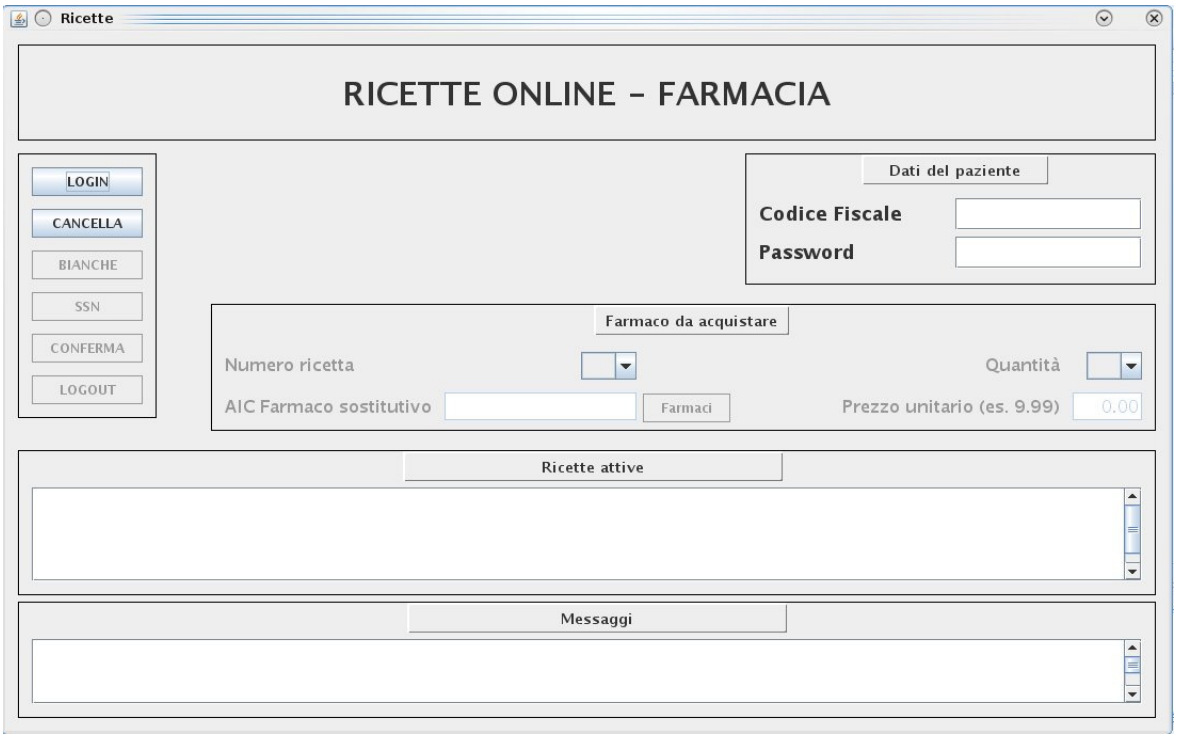

Figura 19: Interfaccia del farmacista prima del login

Sul lato sinistro sono raggruppati i pulsanti del menù principale, alcuni di questi non sono attivi perché si attiveranno solo dopo il login; i pulsanti sono:

- **LOGIN:** il farmacista accede al sistema fornendo le credenziali del paziente:
  - Codice Fiscale
  - Password
- **CANCELLA:** cancella i campi del codice fiscale e password
- **BIANCHE:** consente di elencare le ricette bianche attive per quel paziente; per attive si intende che la sua scadenza sia odierna o posteriore e che le confezioni già acquistate per quel farmaco siano inferiori a quelle prescritte

- ❖ **SSN:** consente di elencare le ricette del SSN attive per quel paziente
- ❖ **CONFERMA:** darà la conferma per l'acquisto del farmaco che il paziente ha deciso di prendere
- ❖ **LOGOUT:** uscita dall'account del paziente

La console centrale è formata da due form uno sopra l'altro, il primo attivo, il secondo no.
Il primo riguarda i *"Dati del paziente"*:

- **Codice fiscale**
- **Password**

Questa parte sarà disattivata non appena il login avrà successo.
Sotto, nel form intitolato *"Farmaco da acquistare"*, c'è l'area relativa al farmaco che deve essere ritirato e si attiverà contemporaneamente alla disattivazione dell'altra:

- **Numero ricetta:** numero d'ordine della ricetta che si vuole prendere in considerazione
- **Quantità:** numero di confezioni che il paziente intende prelevare per quel determinato farmaco
- **AIC Farmaco sostitutivo:** nel caso sia possibile cambiarlo, indica il codice AIC del farmaco che sostituirà quello prescritto
- **Prezzo unitario:** prezzo unitario del farmaco prelevato

Ancora sotto c'è una area di testo, *"Ricette attive"*, che conterrà l'elenco delle ricette bianche o del SSN attive per quel determinato paziente.
Infine l'ultima text area contiene i *"Messaggi"* del sistema, sia quelli di successo che quelli di errore.
La prima operazione che deve compiere il farmacista sarà quella di inserire le credenziali del paziente per loggarsi al sistema.

A questo punto se l'accesso è andato a buon fine può scegliere se:

- Visualizzare le ricette bianche attive per quel paziente premendo **"BIANCHE"**
- Visualizzare le ricette del SSN attive per quel paziente premendo **"SSN"**
- Se ci sono ricette attive per quella tipologia, scegliere quale ricetta e procedere con l'acquisto delle confezioni per quel farmaco premendo **"CONFERMA"**
- Uscire dal sistema premendo **"LOGOUT"**

Per effettuare il login c'è sempre un controllo lato client sul pattern di codice fiscale e password e lato server sull'esistenza dell'associazione tra codice fiscale e password.

6.4.1 Acquistare farmaci dalla ricetta bianca

Dopo aver effettuato il login il farmacista ha davanti l'interfaccia che si trova in questo nuovo stato.

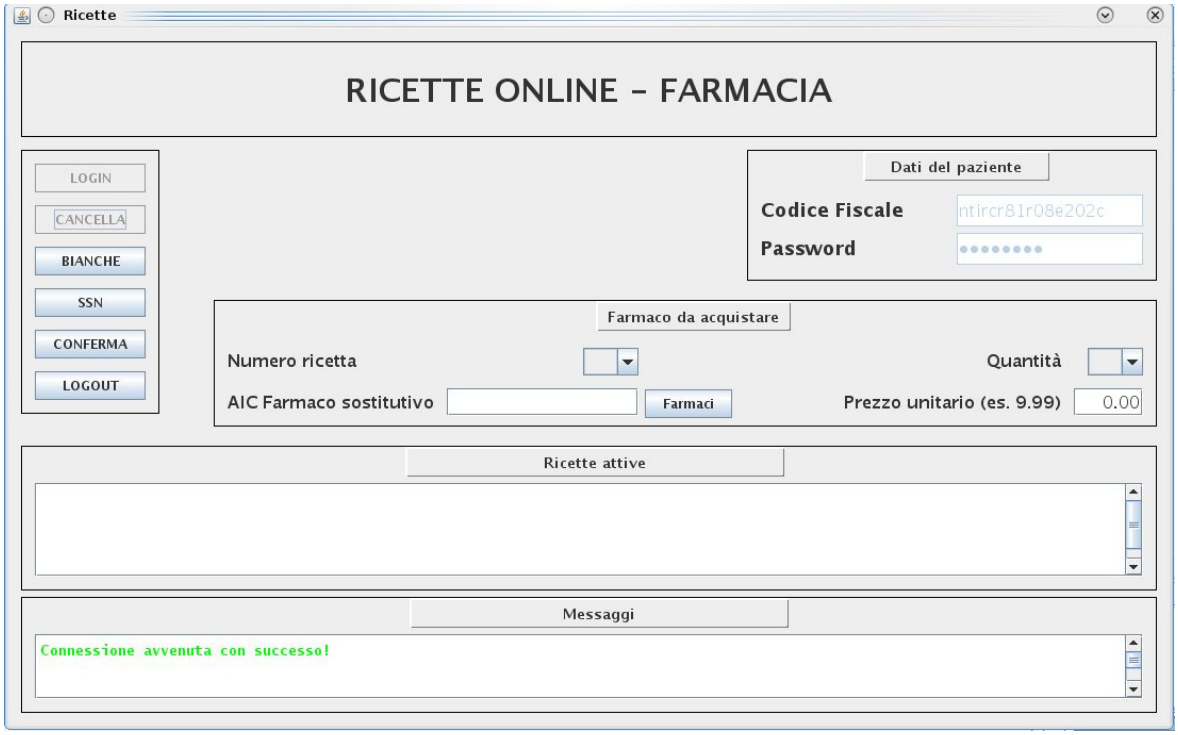

Figura 20: Interfaccia del farmacista a login avvenuto

I campi e i pulsanti relativi all'inserimento delle credenziali sono diventati inattivi, al contrario di quelli riguardanti le ricette.
Premendo il tasto **"BIANCHE"**, nel caso in cui ci siano ricette bianche attive, viene visualizzato un elenco come il seguente.

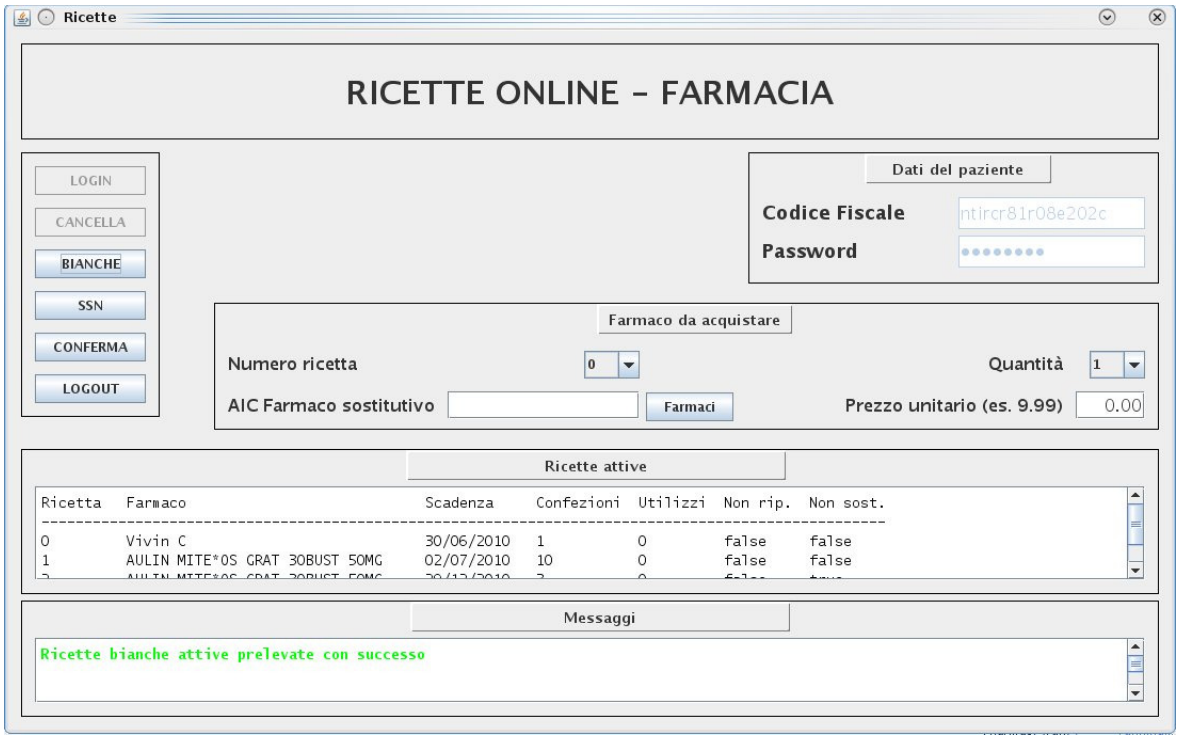

Figura 21: Interfaccia del farmacista che elenca le ricette bianche attive

I campi elencati sono:

- **Ricetta**: numero d'ordine della ricetta
- **Farmaco**: nome comune del farmaco
- **Scadenza**: scadenza per l'acquisto di quel farmaco
- **Confezioni**: numero di confezioni prescritte
- **Utilizzi**: numero di confezioni già acquistate
- **Non rip.**: indica se la ricetta è non ripetibile o meno
- **Non sost.**: indica se la ricetta è non sostituibile o meno

A questo punto procederà con la vendita dei farmaci compilando la parte *"Farmaco da acquistare"* e premendo **"CONFERMA"**. Se non ci sono stati errori nella compilazione del form e l'acquisto è andato a buon fine,

un messaggio di successo confermerà che la transazione è avvenuta; form ed elenco delle ricette saranno ripuliti.
Selezionando una determinata ricetta automaticamente viene settato il campo quantità dei soli valori possibili, ovvero fino alla differenza tra confezioni prescritte e quelle già acquistate. Nel caso in cui nel magazzino non ci siano abbastanza confezioni rispetto a quelle richieste, nel campo dei messaggi vengono elencati i farmaci con lo stesso ATC di quello prescritto e la loro disponibilità.

6.4.2 Acquistare farmaci dalla ricetta del SSN

Premendo il tasto SSN, se presenti, vengono elencate le ricette del SSN attive per quel paziente.

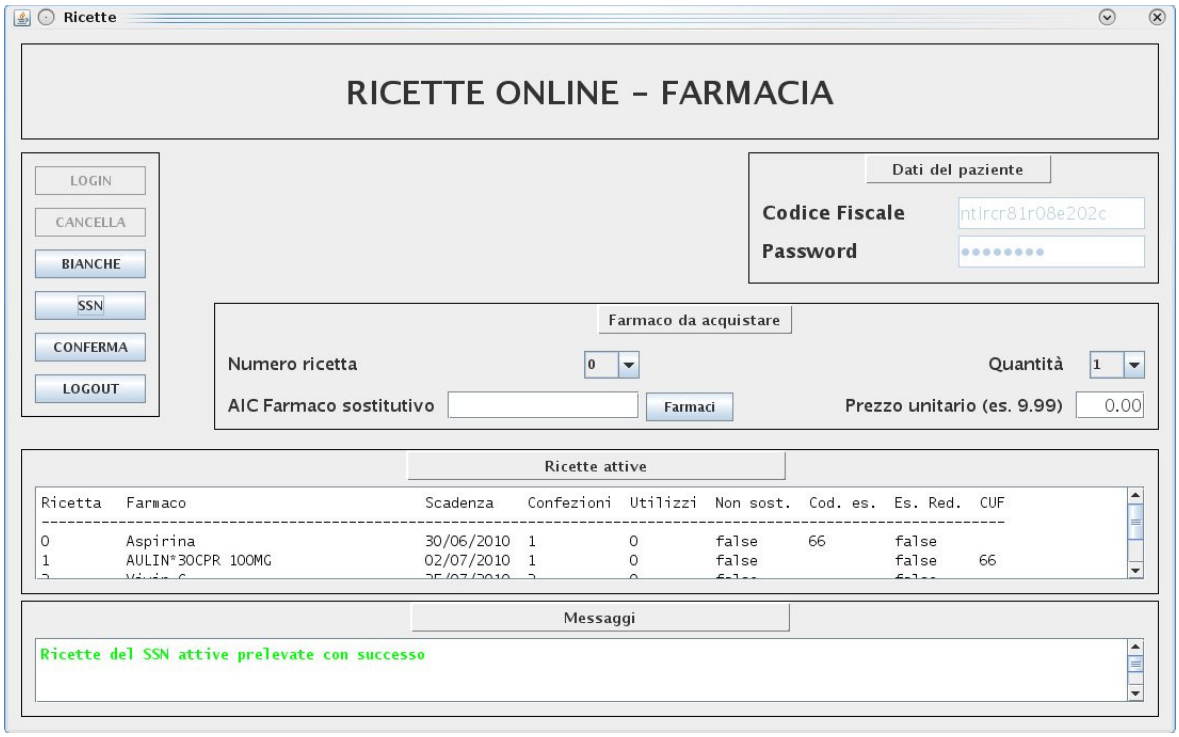

Figura 22: Interfaccia farmacista che elenca le ricette del SSN attive

I campi elencati sono:

- **Ricetta**: numero d'ordine della ricetta
- **Farmaco**: nome comune del farmaco
- **Scadenza**: scadenza per l'acquisto di quel farmaco

- **Confezioni**: numero di confezioni prescritte
- **Non sost.**: indica se la ricetta è non sostituibile o meno
- **Cod. es.**: codice di esenzione del paziente
- **Es. Red.**: indica se il paziente è esente per reddito oppure no
- **CUF**: nota cuf

A questo punto procederà con la vendita dei farmaci compilando la parte *"Farmaco da acquistare"* e premendo **"CONFERMA"**. Se non ci sono stati errori nella compilazione del form e l'acquisto è andato a buon fine, un messaggio di successo confermerà che la transazione è avvenuta; form ed elenco delle ricette saranno ripuliti. Ci sono gli stessi controlli visti nell'altro tipo di ricetta e come prima nel caso in cui nel magazzino non ci siano abbastanza confezioni rispetto a quelle richieste, nel campo dei messaggi vengono elencati i farmaci con lo stesso ATC di quello prescritto e la loro disponibilità.

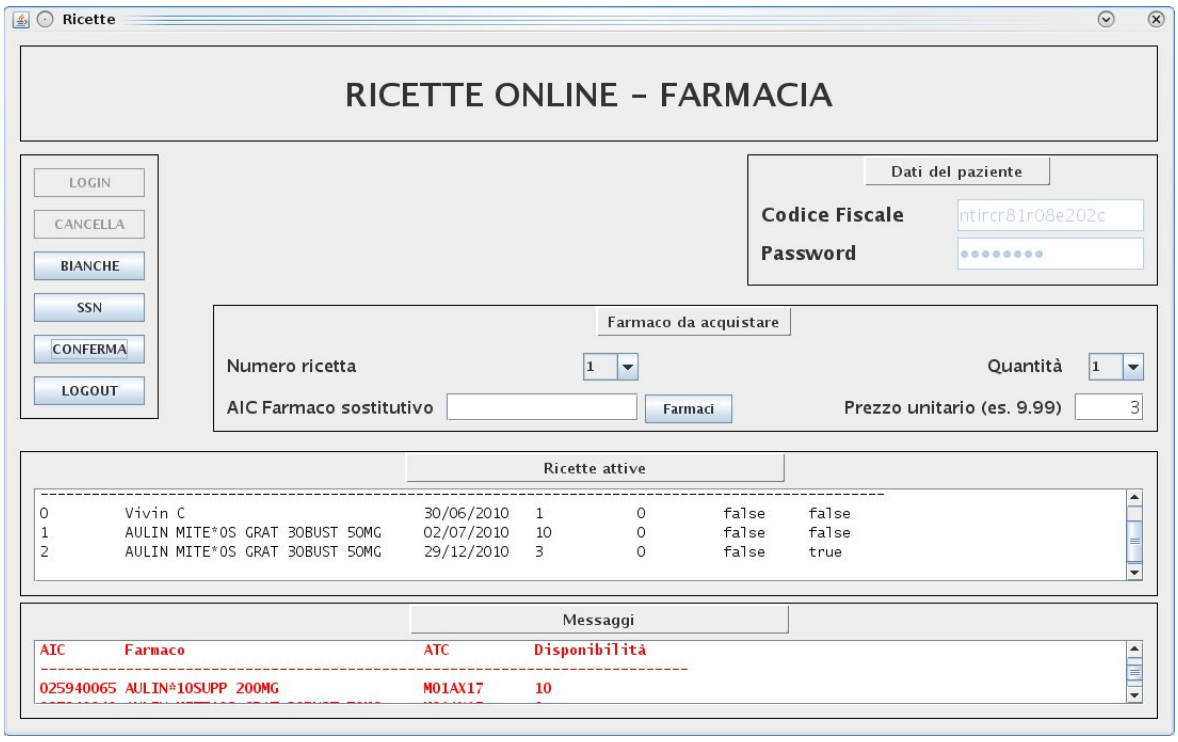

Figura 23: Interfaccia del farmacista che suggerisce farmaci simili

7. Classi e funzioni

Il sistema è composto da cinque progetti indipendenti che collaborano tra di loro:

- ASLServer
- ASLDoctor
- PatientServer
- PharmacyServer
- Pharmacy

Passiamo adesso ad analizzarli uno per uno, illustrando le classi principali ed i metodi più significativi presenti.

7.1 ASLServer

ASLServer è il componente principale del sistema, quello che fornisce servizi a tutti gli altri soggetti; per questa ragione fin da subito si è pensato di dividere le interfacce in base al destinatario:

- ***doctorService.java:*** è l'interfaccia per il programma del dottore
- ***patientService.java:*** è l'interfaccia per il programma del paziente
- ***prescriptionService.java:*** è l'interfaccia per il programma del farmacista

Ad ogni interfaccia corrisponde una classe java che la implementa:

- ***doctorServiceImpl.java***
- ***patientServiceImpl.java***
- ***prescriptionServiceImpl.java***

Nei sottoparagrafi seguenti andremo ad analizzare i metodi che queste mettono a disposizione.

7.1.1 Servizi per il medico

I metodi messi a disposizione del programma del medico sono:

public void login(String doctorCode, String doctorPW) throws CredentialException, GenericException: il dottore fornendo il proprio codice fiscale e la sua password accede al sistema. In caso di credenziali non riconosciute sarà lanciata una eccezione di tipo CredentialException. Nel caso in cui il login abbia successo, viene settato un identificatore di sessione e impostato nei pacchetti SOAP. Questo sarà poi presente nei pacchetti SOAP inviati dal client per confrontarlo con quello memorizzato lato server, tutto ciò allo scopo di validare le richieste del client. Contemporaneamente viene fatto partire un timer che allo scadere del quale la sessione sarà annullata. Il timer viene riavviato ogni volta che c'è una richiesta da parte del client.

public boolean registerPatient(Patient p)throws GenericException: dopo aver controllato la correttezza della sessione, fornendo un oggetto di tipo Patient il dottore procede alla registrazione di un nuovo paziente all'interno del sistema. L'oggetto Patient contiene quelle informazioni necessarie alla registrazione come nome, cognome, codice fiscale e password. Il controllo sulla validità della sessione viene effettuata in tutti i metodi seguenti.

public DetailedPrescription endingPrescriptionBis() throws GenericException, Exception: attraverso questa funzione il dottore recupera tutte le ricette di tipo "bianco" che sono in scadenza oggi e che riguardano tutti i suoi pazienti.

public DetailedSSNPrescription endingSSNPrescriptionBis() throws GenericException, Exception: allo stesso modo con questa funzione recupera tutte le ricette del SSN che sono in scadenza oggi e che riguardano tutti i suoi pazienti.

public boolean insertPrescriptionBis(WhitePrescriptionBis wP) throws GenericException: questa funzione prendendo in ingresso un oggetto di tipo WhitePrescriptionBis, che contiene i dettagli di una ricetta bianca, registra una nuova ricetta per un determinato paziente.

public boolean insertSSNPrescriptionBis(SsnPrescriptionBis sP) throws GenericException: questa è la funzione duale della precedente, per inserire una nuova ricetta di tipo SSN.

public boolean renewPrescription(WhitePrescriptionBis wP, String prescId, String medId) throws GenericException: questa funzione serve per rinnovare una ricetta bianca che sta scadendo oggi. Oltre ai dati della nuova ricetta, contenuti in un oggetto WhitePrescriptionBis, devono essere forniti anche l'identificativo della ricetta che sarà rinnovata e quindi annullata e l'identificativo del medico.

public boolean renewSSNPrescription(SsnPrescriptionBis ssnP, String prescId, String medId) throws GenericException: stessa cosa, ma per le ricette del SSN.

public void closeSession(): il programma dottore chiama questa funzione quando spontaneamente vuole chiudere la sessione in corso. Solo effettuando di nuovo il login e dando il via ad una nuova sessione sarà possibile usufruire i nuovo dei servizi.

7.1.2 Servizi per il farmacista

I metodi messi a disposizione per il programma del farmacista sono:

public void login(String patientCode, String patientPW)throws CredentialException, GenericException: anche per quanto riguarda il programma che gira sul computer del farmacista il primo metodo da invocare sarà questo di login. Inviando il codice fiscale del paziente e la

sua password corrispondente si accederà al sistema. Anche in questo caso viene generato un identificativo random che identifica la sessione appena iniziata. Un timer si occuperà di resettare la sessione se non arrivano richieste entro un certo timeout.

public PrescriptionList getPrescriptionBis(String patientCode) throws GenericException: fornendo il codice fiscale del paziente si ottengono tutte le ricette bianche attive per quel paziente. L'oggetto PrescriptionList al suo interno contiene tutta una serie di vettori, uno per ogni caratteristica significativa della ricetta. Il metodo non va a buon fine se il controllo della sessione non dà esito positivo.

public SSNPrescriptionList getSSNPrescription(String patientCode) throws GenericException: è la funzione duale della precedente, per prelevare le ricette del SSN attive per il paziente indicato.

public boolean getMedicineBis(String pharmacyId, String prescriptionId, String prMedicineId, int amount, String medicineId, String price)throws GenericException: è il metodo usato per indicare l'acquisto di un farmaco presente su una ricetta bianca. Gli argomenti della funzione sono:

- Identificativo della farmacia in cui avviene l'acquisto
- Identificativo della ricetta
- Farmaco prescritto
- Numero di confezioni da acquistare
- Farmaco sostitutivo
- Prezzo unitario della confezione

public boolean getSSNMedicine(String pharmacyId, String prescriptionId, String prMedicineId, int amount, String medicineId, String price) throws GenericException: stessa funzione, ma per un medicinale prescritto su una ricetta del SSN.

public void closeSession(): metodo usato dal programma del farmacista per chiudere volontariamente la sessione. Ogni altra richiesta non verrà accettata fino ad un nuovo login.

7.1.3 Servizi per il paziente

Simili sono i metodi messi a disposizione per il web service che gestisce la comunicazione con il browser del paziente:

public void login(String patientCode, String patientPW)throws CredentialException, GenericException: codice fiscale e password che il paziente inserisce nel form di autenticazione sono inviati a questa funzione. Se l'utente viene riconosciuto si dà il via ad una nuova sessione.

PrescriptionList getPrescription(String patientCode)throws GenericException: una volta entrato nel sistema il paziente richiede la visualizzazione di delle ricette bianche attive attraverso questa funzione.

SSNPrescriptionList getSSNPrescription(String patientCode) throws GenericException: questa funzione invece serve per raccogliere le informazioni riguardanti le ricette del SSN attive.

7.1.4 Package DB

Il package DB raccoglie tutte quelle classi e quelle strutture dati utilizzate dai precedenti metodi per descrivere le ricette. In particolar modo il file ***DBManager.java***, utilizzando la libreria JDBC (Java DataBase Connectivity), è la classe demandata a tutte le interazioni con il database presente sul server della ASL. Questa classe contiene tutte le informazioni e le credenziali per l'accesso al database ed i metodi per interrogare il database. Questa classe viene utilizzata da tutte quelle che implementano le interfacce ed è il vero cuore del servizio.

7.1.5 Package sessionPackage

Il package sessionPackage contiene il file ***sessionManager.java*** demandato alla gestione delle sessioni. Quando l'utente effettua il login, se l'operazione ha successo viene chiamato il metodo per settare una nuovo session id e viene instanziato un thread che sarà fatto partire ad intervalli regolari. Ad ogni nuova richiesta di servizio, verrà invocata la funzione per controllare se il session id inviato nella richiesta dal client corrisponde a quello memorizzato sul server. Se il confronto dà esito positivo il thread dormiente viene eliminato e ne viene creato un altro. Allo scadere del timeout il thread partirà con la propria esecuzione e resetterà la sessione azzerando il session id. Richieste successive da parte del client non saranno accettate. Questo sistema è stato realizzato per evitare che una disconnessione improvvisa del client o comunque un periodo di inattività troppo prolungato potessero dar vita a sessioni pendenti senza fine e che generassero possibili lacune nella sicurezza dell'intero sistema.

7.1.6 Package exceptionPackage

Questo package raccoglie due tipi di eccezioni lanciate dai vari metodi. Le due tipologie non hanno niente di particolare, ma ho scelto di creare due tipi nuovi per poter facilmente identificare la causa che ha generato tale eccezione:

- *CredentialException:* viene lanciata nei casi in cui non vengano riconosciute le credenziali fornite per il login
- *GenericException:* raccoglie tutti gli altri tipi di eccezioni, la causa più specifica deve essere ricercata nel campo dettaglio al suo interno

7.2 ASLDoctor

ASLDoctor è il primo client che utilizza i servizi messi a disposizione dal server della ASL.

Per la realizzazione ho cercato di seguire il pattern Model View Control per ottenere una ripartizione intelligente dei compiti tra i vari package o classi. Questo progetto è composto da due package oltre quello creato per aver importato il file WSDL del server:

- *guipackage*
- *asldocclient*

Li analizzeremo velocemente uno di seguito all'altro.

7.2.1 Package guipackage

Il guipackage contiene al suo interno i file che implementano le interfacce grafiche utilizzate. Tutte le interfacce estendono la classe *javax.swing.JFrame*:

***GUI.java**:* è l'interfaccia grafica principale del programma, è quella visualizzata all'avvio del programma, grazie a questa si possono inserire le nuove ricette o far partire le interfacce per le altre azioni possibili.

***regPatGUI.java**:* fornisce l'interfaccia per registrare un nuovo paziente all'interno del sistema.

***EndingPrescrGUI.java**:* realizza l'interfaccia grafica lanciata dal dottore quando vuol sapere quali sono le ricette mediche in scadenza nella giornata odierna per tutti i suoi pazienti.

7.2.2 Package asldocclient

Questo package contiene un solo file ***ASLDocClient.java*** che ha la funzione di essere l'endpoint lato client per invocare i servizi sul server. Riporto di seguito parti significative del codice di questa classe.

```
// Endpoint for SSL connection
   private static final String endpoint =
         "https://localhost:8443/ASLServer/doctorServiceImpl";
```

```
// Service endpoint
private service.DoctorServiceImplService service = null;
private service.DoctorService port = null;
```

La stringa *endpoint* memorizza l'url a cui il client si collegherà mentre invece le variabili *service* e *port* conterranno il punto di accesso per invocare i metodi remoti. Da notare che l'endpoint rappresenta un indirizzo "https", infatti tutte le comunicazioni sono cifrate grazie a SSL.

public void checkDocCredential(String username, String password) throws CredentialException_Exception, GenericException_Exception utilizzato per accedere al sistema fornendo le credenziali del dottore, prima di chiamare il metodo remoto fa alcune operazioni molto importanti.

```
System.setProperty("javax.net.ssl.trustStore","/home/memphis/.keystore");

Map<String, Object> req_ctx = ((BindingProvider) port).getRequestContext();
req_ctx.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, endpoint);
```

Con la prima indica il path per il truststore che conserva i certificati accettati e con le successive indica che l'url di destinazione delle richieste sarà quello indicato nella variabile *endpoint.*
Il codice seguente invece serve per conservare le sessioni dal lato client. In questo modo ogni volta che effettuerà una richiesta per un servizio, nel suo pacchetto sarà trasportato anche il session id settato dal server.

```
((BindingProvider)port).getRequestContext().put(BindingProvider.SESSION_MAI
NTAIN_PROPERTY,true);
```

All'interno di questa classe ci sono anche due funzioni per la gestione delle date di scadenza:

public Calendar calcExDate(boolean white, boolean not_rep): dato il tipo di ricetta e, nel caso in cui sia bianca, la sua ripetibilità o meno, calcola la scadenza limite che quella ricetta può avere.

public boolean checkPrescrDate(boolean white, boolean not_rep, String expDate): in base alla data di scadenza immessa e alle caratteristiche della ricetta indica se la scadenza indicata è valida, ovvero compresa tra la data odierna e quella limite.

7.3 PatientServer

PatientServer è un client rispetto al server della ASL, ma a sua volta è un server nei confronti del browser web del paziente. Utilizzando i servizi di interrogazione del database delle ricette, permette ai pazienti di visualizzare le proprie ricette attive attraverso una interfaccia web.
La cartella relativa al sito internet contiene tutta una serie di file html e jsp che costituiscono le pagine di navigazione:

- ***index.jsp***: è la pagina iniziale, quella che viene visualizzata quando si digita la URL del server. Al suo interno contiene solamente il form utilizzato per raccogliere i dati del paziente ed inviarli ad un'altra pagina.

- ***send.jsp***: raccoglie i dati inviatogli dalla pagina index e crea un oggetto di tipo patClient che fa da punto di collegamento con i servizi offerti dal server della ASL e lancia la funzione login.

```
patientServerClient.patClient cl = new patientServerClient.patClient();
String sessionId = cl.login(request.getParameter("patientCode"),
                            request.getParameter("patientPW"));
```

  Se le credenziali sono state riconosciute restituisce un valore che identificherà la sessione e farà un redirect alla pagina principale di navigazione. Questa stringa verrà memorizzata come variabile di sessione ed accodata in tutte le URL durante la navigazione. Ogni

pagina successiva per accertarsi che l'utente si sia già connesso farà un confronto tra la stringa dell'URL e quella memorizzata.

session.setAttribute("SID", sessionId);
response.sendRedirect("./home.jsp?SID="+sessionId);
Come variabile di sessione viene memorizzato anche il codice fiscale il quale sarà utilizzato per identificare rispetto a quale paziente si vogliono ricevere informazioni sulle ricette

session.setAttribute("patCode", request.getParameter("patientCode"));

- ***home.jsp***: è il contenitore per la pagina di navigazione principale. E' composto da tre frame, uno in alto per il titolo, uno laterale per il menù ed uno centrale per contenere le pagine lanciate dal menù.

- ***header.html***: è il titolo della pagina di home

- ***menu.jsp***: costituisce il frame utilizzato per contenere il menù. Si può vedere come all'URL delle pagine linkate sia aggiunto il session id.

<a href="./white.jsp?SID=<%=sid%>" target="center">Ricette bianche attive
<a href="./ssn.jsp?SID=<%=sid%>" target="center">Ricette del SSN attive

- ***welcome.html***: contiene un messaggio per confermare l'accesso al sistema con successo

- ***white.jsp***: questa pagina serve per visualizzare le ricette bianche attive. Prima di procedere fa un confronto per verificare la correttezza della sessione.

String urlSID = request.getParameter("SID");
String sessSID = (String)session.getAttribute("SID");

```
// Confronto sid
if(urlSID.equals(sessSID))
    {…
```

- ***ssn.jsp***: allo stesso modo si comporta questo file, che ha la funzione di visualizzare le ricette del SSN

- ***./css/webStyle.css***: questo file CSS contiene gli stili applicati a tutti i file precedenti

La parte relativa alle classi java contiene due package.

7.3.1 Package patientServerClient

Tale package contiene il file ***patClient.java*** che rappresenta il punto di attacco lato client con i servizi messi a disposizione dal server della ASL. I metodi contenuti in questa classe non fanno altro che richiamare i metodi remoti del server.

7.3.2 Package sessionPackage

Il sessionPackage contiene il file ***sessionManager.java*** questa classe ha solo il metodo ***public String setSessionId()*** che genera un numero random da usare come session id.

7.4 PharmacyServer

Questo server gira in una macchina nei locali della farmacia ed ha il compito di gestire il magazzino. Come il server della ASL ci sono due file separati riguardo ai servizi offerti:

- ***PharmacyService.java****:* rappresenta l'interfaccia
- ***PharmacyServiceImpl.java****:* implementa i servizi descritti nell'interfaccia

7.4.1 Servizi per il farmacista

Di seguito i servizi che mette a disposizione:

public boolean checkMedicine(String medicineId): prendendo in ingresso il codice AIC del farmaco consulta il database e vede se questo farmaco è in elenco o meno.

public int getQuantity(String medicineId) throws PharmacyException: indicando il codice AIC per un farmaco restituisce il numero di confezioni presenti in magazzino.

public boolean getMedicine(String medicineId, int amount) throws PharmacyExceptio:al momento dell'acquisto di una certa quantità di un farmaco, aggiorna il magazzino, riducendo di *amount*, la quantità di pacchetti presenti.

public String getAtc(String medicineId) throws PharmacyException: restituisce la categoria ATC di appartenenza di un farmaco; questa informazione sarà utilizzata per trovare farmaci simili.

public MedicineList getSimilar(String atc)throws PharmacyException: indicando il codice ATC compila la struttura dati *MedicineList* con tutti i farmaci presenti in magazzino e appartenenti alla stessa categoria ATC.

7.4.2 Package database

Contiene il file ***Manager.java*** che è la classe che si occupa dell'interazione con il database del magazzino della farmacia e la classe ***MedicineList.java***. Questa seconda classe serve come contenitore per la lista di medicinali con stesso ATC.

Le informazioni contenute per ciascun farmaco sono:

- *medicineId:* codice AIC del farmaco
- *medicineName:* nome del farmaco
- *atc:* codice ATC del farmaco
- *available:* numero di confezioni disponibili per quel farmaco

7.4.3 Package exceptionpackage

Anche per il server presente sul computer della farmacia ho creato una classe specifica per le eccezioni che genera in modo che il programma client possa identificare immediatamente da dove proviene quella eccezione. Il campo dettaglio, in seguito, aiuterà ad identificare la causa precisa che la ha generata.

7.5 Pharmacy

Il programma usato dal farmacista è client sia per i servizi offerti dal server della ASL, sia delle funzioni per la gestione del magazzino fornite dal server presente in farmacia o nei locali del magazzino della stessa.

7.5.1 Package guipackage

Il guipackage contiene il file ***GUI.java*** che derivando dalla classe javax.swing.JFrame implementa l'interfaccia grafica del programma usato dal farmacista. Al suo interno sono presenti tutti i componenti grafici che consentono l'immissione e la visualizzazione delle informazioni. Al suo interno sono presenti tutte le funzioni che riguardano l'aspetto grafico dell'applicazione, riporto di seguito le più significative:

private int getOptions(int selection): questa funzione in base al tipo di ricetta selezionato ed all'indice della ricetta tra quelle attive, ricava quale è il massimo numero di confezioni acquistabili facendo la differenza tra quelle prescritte e quelle già acquistate.

private void setSelectionView(): una volta effettuato il login viene lanciata questa funzione per disattivare quei campi del form non più utili e di contro attivare quelli necessari per le prossime azioni.

private void printConfirmMsg(String msg): questa è la funzione per scrivere un messaggio di successo nell'area riservata ai messaggi; passandogli il testo da scrivere *msg*, questi automaticamente setterà il colore del testo e lo scriverà nell'area.

private void printWhitePrescr(): questa funzione viene chiamata per stampare le informazioni delle ricette bianche attive per quel determinato paziente. Prima di stampare fa un controllo per vedere se la lista non è vuota.

private void printSSNPrescr(): stessa funzione della precedente, ma chiamata per stampare le ricette del SSN.

private void printSimilar(): questa funzione invece stampa i farmaci simili, con stesso ATC.

7.5.2 Package aslserverclient

La classe presente nel package è *ASLClient.*
Questa classe è il punto di attacco con il server della ASL, i suoi metodi non fanno altro che richiamare i metodi remoti del server. Al suo interno sono presenti l'endpoint e tutti quei settaggi per poter rendere la comunicazione protetta attraverso l'uso di SSL.
Anche in questo caso deve essere settato a *true* il parametro *BindingProvider.SESSION_MAINTAIN_PROPERTY*.
Di seguito riporto alcune funzioni significative:

private boolean checkChangeable(String medicine, boolean notChangeable): controllando la sostituibilità del farmaco,

notChangeable, e l'eventuale tentativo di cambiare il farmaco, *medicine* diverso da **null**, restituisce la fattibilità o meno del cambio.

private boolean checkRepeatability(int amount, int available, boolean notRepeatability): controlla che, nel caso di ricetta non ripetibile, la quantità di confezioni che si intende acquistare sia uguale a quelle prescritte.

7.5.3 Package pharmacyclient

Questo package contiene il file ***PharmClient.java*** che fa da punto di collegamento con i servizi offerti dal server del magazzino della farmacia. I metodi presenti sono corrispondenti a quelli presenti sul server. In questo caso non si è ritenuto necessario proteggere la comunicazione tramite SSL in quanto si è ipotizzato che server del magazzino e client possano essere in esecuzione nella stessa rete, se non sullo stesso computer.

7.5.4 Package configuration

Per evitare di immetter ogni volta che si avvia il programma il codice identificativo della farmacia, ho aggiunto questo package, al cui interno ci sono:

conf.xml: file di tipo XML che, indicato dal tag <*PHARMACY_ID*>, conserva l'identificativo univoco della farmacia. Questo dato è molto importante per il rimborso dei farmaci forniti dal SSN e viene memorizzato nel database della ASL al momento della vendita del farmaco.

ConfigurationManager.java: è la classe usata per effettuare il parsing del file di configurazione. L'unico metodo presente al suo interno è ***public String getPharmacyId()*** che serve proprio per ricavare l'identificativo della farmacia.

8. Database

Tutte le informazioni scambiate ed aggiornate all'interno del sistema sono memorizzate in una base di dati, per la precisione in due tipi di database:

- **Database della ASL (asl)**
- **Database per il magazzino della farmacia (farmacia)**

8.1 Database della ASL

Il database gestito dalla ASL conterrà tutte quelle informazioni necessarie ad espletare i servizi verso i tre tipi di client. Nei paragrafi seguenti presenterò le varie tabelle con i loro campi, la chiave primaria ed il loro significato.

8.1.1 Tabella dottore

La tabella dottore contiene i dati anagrafici dei dottori che fanno capo a quella ASL.
I campi della tabella sono:

- id_dottore(Primary Key): contiene l'identificatore univoco del dottore
- codFiscale: codice fiscale del dottore
- nome
- cognome
- password: password per l'accesso al sistema

8.1.2 Tabella farmacia

Contiene tutti i dati relative alle farmacie che aderiscono a questo sistema di ricette on line, i campi presenti sono:

- id_farmacia(Primary Key): identificatore univoco della farmacia
- nome: nome commerciale della farmacia
- indirizzo
- città

8.1.3 Tabella paziente

La tabella paziente contiene i dati anagrafici dei pazienti registrati sotto quella ASL. Per adesso sono stati inseriti solo i dati strettamente necessari, ma in sviluppi futuri sarebbe interessante aggiungere molte altre informazioni per esempio i tipi di esenzioni a cui ha diritto il paziente.
Le informazioni contenute sono:

- codFiscale (Primary Key): identificatore univoco del paziente
- cognome
- nome
- pw_paziente
- dottore: contiene l'identificatore di un medico per indicare quale è il medico curante di quel paziente

8.1.4 Tabella farmaco

Contiene la lista di tutti i farmaci in commercio. I campi presenti sono:

- id_farmaco (Primary Key): codice AIC del farmaco
- nome: nome commerciale del farmaco
- categoria: categoria del farmaco secondo il regime di rimborsabilità

Anche in questo caso, in sviluppi futuri l'aggiunta di informazioni più specifiche darebbe la possibilità di realizzare servizi molto potenti.

8.1.5 Tabella ricetta

La tabella "ricetta" contiene le informazioni generali relative alle ricette bianche prescritte:

- id_ricetta (Primary Key): identificativo della ricetta
- data: data di creazione della ricetta
- dottore: specifica il codice del dottore che ha prescritto la ricetta
- paziente: contiene il codice del paziente destinatario

8.1.6 Tabella dettaglio_ricetta

Dato che tra ricetta e farmaci prescritti ci può essere un rapporto uno a molti, ho utilizzato questa tabella separata per specificarne i dettagli.
Questa tabella descrive i farmaci presenti nella ricetta bianca attraverso i seguenti campi:

- id_dettaglio (Primary Key): identifica la tupla del dettaglio
- ricetta: specifica a quale ricetta fa riferimento
- scadenza: scadenza per l'acquisto del farmaco
- farmaco: codice AIC del farmaco
- confezioni: numero di confezioni prescritte
- utilizzi: numero di confezioni già acquistate
- non_sostituibile: indica la clausola di non sostituibilità del farmaco in fase di acquisto
- non_ripetibile: indica la clausola di non ripetibilità della ricetta
- somministrazione: indica le modalità di somministrazione del farmaco
- rinnovata: è un campo che serve al sistema per verificare se quella ricetta è stata rinnovata dalla creazione di un'altra con scadenza posticipata

8.1.7 Tabella ricetta_ssn

Ho deciso di tenere separate le informazioni relative hai due tipi di ricette per semplificare l'inserimento e la consultazione delle

informazioni. Questa tabella contiene le informazioni generali sulla ricetta del SSN attraverso i seguenti campi:

- id_ricetta (Primary Key): identificativo della ricetta
- data: data in cui la ricetta è stata compilata
- dottore: codice del dottore che ha scritto la ricetta
- paziente: codice fiscale del paziente destinatario della ricetta
- cod_esenzione: eventuale codice di esenzione associato al paziente
- reddito: flag per indicare se il paziente ha diritto all'esenzione per reddito

8.1.8 Tabella dettaglio_ssn

Anche in questo caso ho deciso di specificare i farmaci contenuti nella ricetta utilizzando un'altra tabella. In questo caso il rapporto tra ricetta dal SSN e farmaci elencati può essere fino a quello di uno a due.
Le informazioni registrate sono.

- id_dettagli_ssn (Primary Key): identificatore univoco del dettaglio
- ricetta: ricetta SSN a cui fa riferimento
- farmaco: AIC del farmaco prescritto
- confezioni: numero di confezioni prescritte
- utilizzi: numero di confezioni acquistate
- scadenza: data di scadenza per effettuare l'acquisto
- non_sostituibile: clausola di non sostituibilità del farmaco
- cuf: nota AIFA relativa al farmaco
- somministrazione: modalità di somministrazione del farmaco
- rinnovata: è un campo che serve al sistema per verificare se quella ricetta è stata rinnovata dalla creazione di un'altra con scadenza posticipata

8.1.9 Tabella farmaci_acquistati

Questa tabella ha lo scopo di tenere traccia dell'acquisto dei farmaci, prescritti in una ricetta bianca, ed in quale farmacia sono stati acquistati.

Aggiungendo l'identificazione univoca della confezione si potrebbe realizzare la tracciabilità farmaco sino al paziente.

- id_farmaci_acquistati (Primary Key): identificatore dell'acquisto
- ricetta: ricetta a cui fa riferimento l'acquisto
- farmaco: farmaco prescritto nella ricetta e che si vuole acquistare
- farmacia: farmacia in cui avviene l'acquisto
- data: data dell'acquisto
- farmaco_sostituto: l'eventuale farmaco sostituto che si è realmente acquistato
- quantità: numero di confezioni acquistate
- prezzo_unitario: prezzo unitario della confezione

8.1.10 Tabella farmaci_acquistati_ssn

E' la tabella che si occupa di conservare le informazioni relative all'acquisto di farmaci prescritti sulla ricetta del SSN. Questa tabella ha ancor più valore della precedente poiché sarà usata per calcolare i rimborsi da corrispondere alle farmacie.
Le informazioni contenute sono:

- id_farmaci_acquistati: identificatore univoco degli acquisti
- ricetta: ricetta a cui fa riferimento
- farmaco: farmaco specifico che è stato acquistato
- farmacia: farmacia che ha venduto il farmaco
- data: data di acquisto
- farm_sostituto: l'eventuale farmaco sostituto che si è realmente acquistato
- quantità: numero di confezioni acquistate
- prezzo unitario: prezzo unitario della confezione

8.2 Database della farmacia

Il database che gira sulle macchine della farmacia ha una sola tabella perché per questo livello di sviluppo del sistema l'unica funzionalità richiesta era la gestione del magazzino.

8.2.1 Tabella magazzino

Come detto precedentemente la tabella “magazzino” ha il solo scopo di fornire informazione sui farmaci in magazzino e sulla loro disponibilità

- id_farmaco(Primary Key): codice AIC del farmaco
- nome: nome commerciale del farmaco
- quantità: numero di confezioni disponibili
- ATC: categoria ATC di appartenenza; è il campo utilizzato per ricavare farmaci con caratteristiche curative simili in caso di mancanza di disponibilità di quello prescritto
- categoria: categoria di appartenenza secondo il regime di rimborsabilità

9. Sviluppi futuri

Durante l'esplorazione di questo campo e l'implementazione di questa soluzione software sono venute alla luce tante migliorie che potrebbero essere realizzate e tanti servizi che potrebbero essere integrati.
Ci sono alcuni miglioramenti che potrebbero essere realizzati con il minimo sforzo pur fornendo un grande incremento di usabilità.
Ad esempio si potrebbe integrare l'uso di un lettore di codici a barre per velocizzare l'inserimento del codice AIC dei farmaci, o un lettore di tessere per leggere direttamente il codice fiscale e la password del paziente.
Un enorme passo avanti potrebbe essere fatto implementando dei servizi che forniscano informazioni riguardo ai farmaci. Una base di dati con le informazioni utili c'è già e può essere consultata sul sito dell'AIFA, ma la consultazione è resa possibile solo attraverso browser e form html.
Le informazioni a cui mi riferisco sono:

- AIC
- ATC
- Regime di fornitura
- Scadenza della prescrizione

Da queste informazioni si potrebbe ricavare un servizio in cui il medico semplicemente indicando un farmaco attraverso il suo nome comune otterrebbe la ricetta compilata in tutte le sue parti, o verrebbe a conoscenza se esistono farmaci simili o generici.
Inoltre aver informazioni più specifiche sui farmaci permetterebbe di trattare tutti quei casi particolari della ricetta e che non sono stati affrontati in questo elaborato di tesi.
Per quanto riguarda il server della farmacia, se rendesse accessibili al pubblico le informazioni del suo magazzino, si potrebbe creare un

servizio in cui il paziente potrebbe cercare quelle farmacie più vicine a lui che hanno a disposizione quel farmaco in quel determinato momento. Anche la sicurezza del sistema dovrebbe essere migliorata visto la sensibilità delle informazioni trasmesse. La Carta Sanitaria Elettronica, attualmente in fase di distribuzione in Toscana, è stata già predisposta per contenere un certificato digitale, il quale potrebbe essere usato per autenticare il paziente, invece dell'utilizzo di codice fiscale e password.

Indice delle figure

Indice delle tabelle

Bibliografia

Kalin M. (2009), *Java Web Services: Up and Running*, O'Reilly Media, Sebastopol

Avvenuti M., Cecchetti G., Cimino M. G. C. A. (2008), *Lezioni e laboratori di tecnologie web lato client*, SEU, Pisa

Chappell D. A. (2002), *Java web services*, Hops, Milano

Holzner S. (2001), *XML: tutto e oltre*, Apogeo, Milano

Stella E., Tamaglini G. (2010), *Web Services: analisi di un problema reale, progettazione e realizzazione attraverso l'uso dei Java Web Services*, Tesi di Laurea Specialistica della Facoltà di Ingegneria di Pisa

Sitografia

Definizione di web service, http://it.wikipedia.org/wiki/Web_service

Definizione di web service, http://en.wikipedia.org/wiki/Web_service

Stack dei web service ,
http://en.wikipedia.org/wiki/Web_services_protocol_stack

Stack dei web service ,
http://www.pacificgrids.com/images/web_services_stack.jpg

SSL,
http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.csqzas.doc/sy10660a.gif

JAX-WS, https://jax-ws.dev.java.net/jax-ws-ea3/docs/annotations.html

Notizia sulla ricetta elettronica, http://notizie.it.msn.com/approfondimento/articolo.aspx?cp-documentid=153028976

La ricetta medica, http://www.salvelocs.it/legislazione%201.htm

Sperimentazione della ricetta telematica in Piemonte, http://www.ilsecoloxix.it/p/basso_piemonte/2010/01/11/AMmCkSHDmedici_ricetta_telematica.shtml

Classificazione dei farmaci,
http://santof58.wordpress.com/2009/02/04/classificazione-dei-farmaci/

Carta Sanitaria Elettronica,
http://www.regione.toscana.it/cartasanitaria/index.html

MySQL, http://dev.mysql.com/

Tomcat, http://tomcat.apache.org/